PR #46647 完整报告

vllm-project/vllm

[Refactor] Move iteration logging to the frontend

合并时间：2026-07-16 08:59

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/46647

执行摘要

- 一句话：将迭代细节日志从 EngineCore 移到前端统一输出
- 推荐动作：建议精读此 PR，尤其是数据流设计：EngineCore -> SchedulerStats -> EngineCoreOutputs -> LoggingStatLogger。其条件启用的开关设计和对多模态指标的模型无关处理方法值得参考。此外，编码器统计预计算位置的选择是一个值得关注的设计权衡。

功能与动机

关联 Issue #34860 提出需要在迭代日志中报告 KV cache 使用率。同时社区讨论建议将日志从 EngineCore 直接发射改为前端统一路径（见 PR body）。该 PR 响应了这一需求，将迭代详情日志迁移到前端 stats 记录体系，并以模型无关的方式支持多模态编码器输出嵌入计数。

实现拆解

1. 新增数据结构：在 vllm/v1/metrics/stats.py 中定义 SchedulerIterationDetails dataclass，包含迭代索引、上下文请求 / token 计数、生成请求 / token 计数、耗时、是否为 dummy、编码器输入输出统计等字段。在 vllm/v1/core/sched/output.py 中定义 ScheduledEncoderInputStats dataclass，记录编码器输入数量和输出嵌入总数。在 SchedulerOutput 中添加 scheduled_encoder_input_stats 字段。
2. 改造 EngineCore 日志点：vllm/v1/engine/core.py 中，将 log_iteration_details 上下文管理器替换为 capture_iteration_details，后者不再直接调用 logger.info，而是构造 SchedulerIterationDetails 并通过 yield 返回给调用者。新增 _attach_iteration_details 方法，将捕获到的详情附着到对应的 EngineCoreOutputs.scheduler_stats 上。新增 _make_iteration_details_stats 工厂方法。
3. 前端日志输出：vllm/v1/metrics/loggers.py 中，添加 _log_iteration_details 和 _log_prefix_for_engine 方法。_log_iteration_details 从 SchedulerStats.iteration_details 读取数据，按统一格式输出日志行（包括 KV cache 使用率和可选的编码器指标）。record 方法在每次统计点调用 _log_iteration_details。
4. 调度器预计算编码器统计：vllm/v1/core/sched/scheduler.py 的 schedule 方法中，当 log_stats 和 enable_logging_iteration_details 均启用时，调用新方法 _make_scheduled_encoder_input_stats 遍历已调度的编码器输入，通过 MultiModalFeatureSpec.mm_position.get_num_embeds() 计算输出嵌入总数。

5. 辅助函数调整：vllm/v1/utils.py 中的 compute_iteration_details 现在消费 SchedulerOutput.scheduled_encoder_input_stats，填充 IterationDetails 的编码器相关字段。
6. 测试覆盖：新增 tests/v1/engine/test_iteration_logging.py 验证 capture_iteration_details 的启用 / 禁用、耗时填充、附着逻辑；扩展 tests/v1/core/test_scheduler.py 测试编码器统计计算及禁用条件；tests/v1/metrics/test_stats.py 中增加序列化测试。所有新增测试默认通过 cpu_test 标记运行。

关键文件：

- vllm/v1/engine/core.py（模块 引擎核心；类别 source；类型 core-logic；符号 capture_iteration_details, _make_iteration_details_stats, _attach_iteration_details）：核心变更：将日志记录点从 EngineCore 迁移到数据捕获，新增 capture_iteration_details 和 _attach_iteration_details。
- vllm/v1/metrics/loggers.py（模块 日志；类别 source；类型 core-logic；符号 _log_prefix_for_engine, _log_iteration_details）：前端日志记录器新增 _log_iteration_details 方法，统一输出迭代详情。
- vllm/v1/core/sched/scheduler.py（模块 调度器；类别 source；类型 core-logic；符号 _make_scheduled_encoder_input_stats）：新增编码器输入统计预计算方法 _make_scheduled_encoder_input_stats。
- vllm/v1/core/sched/output.py（模块 调度输出；类别 source；类型 data-contract；符号 ScheduledEncoderInputStats）：新增 ScheduledEncoderInputStats 数据类型。
- vllm/v1/metrics/stats.py（模块 统计；类别 source；类型 data-contract；符号 SchedulerIterationDetails）：新增 SchedulerIterationDetails 作为日志数据载体。
- tests/v1/engine/test_iteration_logging.py（模块 测试；类别 test；类型 test-coverage；符号 FakeEngineCore, make_iteration_details, test_capture_iteration_details_disabled_without_log_stats, test_capture_iteration_details_fills_elapsed_time）：新增测试文件，全面覆盖捕获和附着逻辑。
- tests/v1/core/test_scheduler.py（模块 测试；类别 test；类型 test-coverage；符号 test_make_scheduled_encoder_input_stats_output_embeddings, test_scheduled_encoder_input_stats_disabled_without_iteration_logging, test_scheduled_encoder_input_stats_disabled_without_log_stats, test_scheduler_stats_route_to_existing_output_client）：增加编码器统计和路由测试。

关键符号：capture_iteration_details, _attach_iteration_details, _make_iteration_details_stats, _log_iteration_details, _log_prefix_for_engine, _make_scheduled_encoder_input_stats, compute_iteration_details

关键源码片段

vllm/v1/engine/core.py

核心变更：将日志记录点从 EngineCore 迁移到数据捕获，新增 capture_iteration_details 和 _attach_iteration_details。

```
@contextmanager
def capture_iteration_details(
    self, scheduler_output: SchedulerOutput | None
) -> Generator[SchedulerIterationDetails | None, None, None]:
    # 检查是否启用日志统计和迭代详情功能
    enable_details = (
        self.vllm_config.observability_config.enable_logging_iteration_details
    )
    if not self.log_stats or not enable_details:
        yield None
        return

    # 跳过 0-token 步骤（让 dummy_batch 包装器记录，避免重复）
    if (
        scheduler_output is not None
        and scheduler_output.total_num_scheduled_tokens == 0
    ):
        yield None
        return

    iteration_index = getattr(self, "_iteration_index", 0)

    # scheduler_output 为 None 表示 DP dummy 迭代
    if scheduler_output is None:
        iteration_details = SchedulerIterationDetails(
            iteration_index=iteration_index,
            num_ctx_requests=0,
            num_ctx_tokens=0,
            num_generation_requests=0,
            num_generation_tokens=0,
            elapsed_ms=0.0,
            is_dummy=True,
        )
    else:
        details = compute_iteration_details(scheduler_output)
        iteration_details = SchedulerIterationDetails(
            iteration_index=iteration_index,
            num_ctx_requests=details.num_ctx_requests,
            num_ctx_tokens=details.num_ctx_tokens,
            num_generation_requests=details.num_generation_requests,
            num_generation_tokens=details.num_generation_tokens,
            elapsed_ms=0.0,
            is_dummy=False,
            num_encoder_inputs=details.num_encoder_inputs,
            num_encoder_output_tokens=details.num_encoder_output_tokens,
        )
```

```
    before = time.monotonic()
    yield iteration_details
    # context manager 退出后填充耗时
    iteration_details.elapsed_ms = (time.monotonic() - before) * 1000
    self._iteration_index = iteration_index + 1
```

vllm/v1/metrics/loggers.py

前端日志记录器新增 _log_iteration_details 方法，统一输出迭代详情。

```
def _log_iteration_details(
    self, scheduler_stats: SchedulerStats, engine_idx: int
) -> None:
    details = scheduler_stats.iteration_details
    if details is None:
        return

    # 构造编码器消息（仅当有编码器输入时）
    encoder_msg = ""
    if details.num_encoder_inputs:
        encoder_msg = (
            f", encoder inputs: {details.num_encoder_inputs}, "
            f"encoder output embeddings: {details.num_encoder_output_tokens}"
        )

    # 使用统一格式输出迭代日志
    logger.info(
        "%sIteration(%d): %d context requests, %d context tokens, "
        "%d generation requests, %d generation tokens, "
        "iteration elapsed time: %.2f ms%s, "
        "GPU KV cache usage: %.1f%%%s",
        self._log_prefix_for_engine(engine_idx),
        details.iteration_index,
        details.num_ctx_requests,
        details.num_ctx_tokens,
        details.num_generation_requests,
        details.num_generation_tokens,
        details.elapsed_ms,
        " (dummy)" if details.is_dummy else "",
        scheduler_stats.kv_cache_usage * 100,
        encoder_msg,
    )
```

tests/v1/engine/test_iteration_logging.py

新增测试文件，全面覆盖捕获和附着逻辑。

```
def test_attach_iteration_details_falls_back_to_client_zero_without_outputs():
    # 构造一个空的 outputs 字典
    iteration_details = make_iteration_details()
    outputs: dict[int, EngineCoreOutputs] = {}
```

```
    # 调用附着方法，应 fallback 到 client 0
    EngineCore._attach_iteration_details(FakeEngineCore(), outputs, iteration_details)

    # 验证 outputs 包含 key 0，并且其 scheduler_stats.iteration_details 正确
    assert set(outputs) == {0}
    assert outputs[0].scheduler_stats is not None
    assert outputs[0].scheduler_stats.iteration_details == iteration_details
```

评论区精华

讨论点 1：上下文管理器风格

Isotr0py 建议保持原有的上下文管理器，而不是拆成 _start_iteration_details / _finish_iteration_details，认为后者不易发现日志区域。作者接受并改回了上下文管理器 capture_iteration_details。

讨论点 2：编码器统计计算位置

Isotr0py 询问能否将 scheduled_encoder_input_stats 的计算移到 compute_iteration_details 统一完成。作者解释：compute_iteration_details 只接收 SchedulerOutput，而 mm_features 信息并未包含在输出中（为避免传递完整请求数据），因此需要在调度器内部利用 self.requests[req_id].mm_features 来计算，所以保留在 _make_scheduled_encoder_input_stats 中。

讨论点 3：缩进正确性

Isotr0py 发现 run_busy_loop 中 stats 创建语句在 capture_iteration_details 上下文管理器之外，质疑是否正确。作者确认是故意设计：elapsed_ms 只有在上下文管理器退出后才由 capture_iteration_details 填充，因此 stats 创建必须放在外部。

- 使用上下文管理器替代 start/finish 分离方法 (design): 作者同意并改回使用 capture_iteration_details 上下文管理器。
- 编码器统计计算位置 (design): 作者解释由于 mm_features 不在 SchedulerOutput 中，需要在调度器内预计算，因此保留在 _make_scheduled_encoder_input_stats 中。
- 缩进正确性 (correctness): 作者确认是故意为之，因为 elapsed_ms 在 context manager 退出后才被填充，所以 stats 创建必须放在上下文外部。

风险与影响

- 风险：
  - 功能默认关闭：新日志行为受 --enable-logging-iteration-details 和 log_stats 双重开关控制，默认不开启，不影响现有用户。
  - 日志格式变化：启用后日志格式包含 GPU KV cache usage 和可选的 encoder inputs/embeddings 字段，可能影响依赖旧格式的日志解析工具。
  - 新增数据依赖：SchedulerOutput 增加了 scheduled_encoder_input_stats 字段，对非多模态场景为 None，内存和序列化开销可忽略。

    - 测试覆盖不足路径：部分测试需要 GPU 和多模态模型，在纯 CPU CI 上无法完整执行，但通过单元测试覆盖了核心逻辑。
- 影响：
    - 用户影响（中低）：目标用户为需要迭代级性能分析的高级用户，通过新增命令行选项开启。日志内容更丰富。
    - 系统影响（低）：仅增加少量数据传递（SchedulerIterationDetails 和 ScheduledEncoderInputStats），无性能冲击。
    - 团队影响（积极）：日志集中在前端 LoggingStatLogger，便于统一管理和扩展，后续增加其他迭代级指标只需扩展数据结构。
    - 风险标记：默认关闭的日志变更，日志格式变化可能影响解析，新增数据传递字段

关联脉络

- PR #34860 [Feature] Add KV *cache* usage metrics to iteration logging: 该 PR 实现了此 issue 中提出的 KV 缓存使用率指标，并将日志路径从 EngineCore 移到前端。
- PR #31193 [Feature] Add iteration level logging: 原始迭代日志 PR，此重构在其基础上扩展并迁移日志路径。