

PR #49984 完整报告

vllm-project/vllm

[Feat] Add request-level preemption count histogram metric

合并时间: 2026-09-01 11:04

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/49984>

执行摘要

- 一句话: 新增请求级抢占次数直方图, 细化抢占可观测性
- 推荐动作: 改动小而清晰, 值得作为 vLLM 指标层入门的精读样例。值得关注的决策点有两个: 一是 histogram 语义设计 (per-request lifecycle 指标与节点级事件计数器的分工), 二是 bucket 边界对 Ray Metrics Exporter 的兼容处理。若运维团队关注 KV cache 压力下的服务质量, 可基于该指标建立 "高抢占次数请求占比" 告警。

功能与动机

PR body 说明目的: "This PR introduces request-level preemption tracking by adding a new `vllm:request_num_preemptions` histogram metric... providing better visibility into system performance and capacity limits under load." 作者在 review 中进一步澄清: 现有 `vllm:num_preemptions` 只统计节点 / 引擎层面的抢占事件总数, 无法回答 "每个请求被抢占了几次、是否集中在少数长上下文请求上"; 直方图能区分均匀抢占 (每请求 1 次) 与严重 thrashing (单请求 5-10+ 次), 并与已有的 per-request 指标体系 (`vllm:request_prompt_tokens`、`vllm:request_queue_time_seconds` 等) 保持一致。

实现拆解

1. 数据源计数: `vllm/v1/metrics/stats.py` 中 `RequestStateStats` 新增 `num_preemptions` 字段; `update_from_events` 在 `EngineCoreEventType.PREEMPTED` 分支内同时执行 `self.num_preempted_reqs += 1` 与 `req_stats.num_preemptions += 1`, 复用现有事件流, 不侵入调度器。
2. 结束信息传递: `FinishedRequestStats` 新增同名字段, `update_from_finished_request` 将 `req_stats.num_preemptions` 写入 `finished` 记录, 随请求完成进入 `finished_requests` 队列。
3. 指标定义与上报: `vllm/v1/metrics/loggers.py` 在 histogram 组中新增 `vllm:request_num_preemptions`, buckets 为 `[1, 2, 3, 4, 5, 10, 20]`, 特意不含 0 以满足 Ray Exporter 对 histogram 边界必须为正的要求; `record()` 遍历 `finished_requests` 时对该直方图 `observe`。
4. 测试配套: `tests/entrypoints/serve/instrumentator/test_metrics.py` 在 `_get_expected_values` 增加 `count` 断言, 并在 `EXPECTED_METRICS` 中登记 `_sum/_bucket/_count` 三个系列的存在性检查。

5. 语义边界：prefill 与 decode 阶段的抢占都累计到同一计数器，指标只统计被抢占次数，不区分阶段。

关键文件：

- vllm/v1/metrics/loggers.py（模块 指标采集；类别 source；类型 core-logic；符号 `Logger.init`, `Logger.record`, `histogram_request_num_preemptions`）：定义并上报新直方图指标，是用户可观测出口；核心改动在 `histogram` 注册与 `finished_requests` 循环中的 `observe`。
- vllm/v1/metrics/stats.py（模块 指标统计；类别 source；类型 core-logic；符号 `RequestStateStats`, `FinishedRequestStats`, `update_from_events`, `update_from_finished_request`）：提供请求级抢占计数的数据来源：在 `PREEMPTED` 事件处累加，并在请求结束时传递给 `FinishedRequestStats`。
- tests/entrypoints/serve/instrumentator/test_metrics.py（模块 指标测试；类别 test；类型 test-coverage；符号 `_get_expected_values`, `EXPECTED_METRICS`）：集成测试登记新指标的 `count` 断言与 `series` 存在性检查，防止新指标被误删或漏报。

关键符号： `Logger.init`, `Logger.record`, `update_from_events`, `update_from_finished_request`

关键源码片段

vllm/v1/metrics/loggers.py

定义并上报新直方图指标，是用户可观测出口；核心改动在 `histogram` 注册与 `finished_requests` 循环中的 `observe`。

```
# vllm/v1/metrics/loggers.py —— 在 __init__ 中登记直方图
histogram_request_num_preemptions = self._histogram_cls(
    name="vllm:request_num_preemptions",
    documentation="Histogram of the number of times a request was preempted.",
    # 刻意不含 0: Ray 对所有 histogram 边界要求为正数
    buckets=[1, 2, 3, 4, 5, 10, 20],
    labelnames=labelnames,
)

# 按 engine 维度包装，与既有 per-request 指标保持一致
self.histogram_request_num_preemptions = create_metric_per_engine(
    histogram_request_num_preemptions, per_engine_labelvalues
)

# 在 record() 的 finished_requests 循环中上报
for finished_request in iteration_stats.finished_requests:
    # 其余 per-request 指标 observe 与此并列，此处只展示新增行
    self.histogram_request_num_preemptions[engine_idx].observe(
        finished_request.num_preemptions
    )
```

vllm/v1/metrics/stats.py

提供请求级抢占计数的数据来源：在 PREEMPTED 事件处累加，并在请求结束时传递给 FinishedRequestStats。

```
# vllm/v1/metrics/stats.py —— 请求级抢占计数的核心逻辑
@dataclass
class RequestStateStats:
    """跨 delta 更新持续跟踪的请求状态。"""
    num_generation_tokens: int = 0
    # 新增：PREEMPTED 事件驱动的每请求次数
    num_preemptions: int = 0
    # 其余时间戳字段略

def update_from_events(self, req_id, events, is_prefilling, req_stats, lora_states, lora_name):
    # 与节点级计数共用同一事件流，确保两种口径一致
    for event in events:
        if event.type == EngineCoreEventType.PREEMPTED:
            self.num_preempted_reqs += 1 # 既有节点级总数
            req_stats.num_preemptions += 1 # 新增请求级计数
            lora_states.request_waiting(req_id, lora_name)

def update_from_finished_request(self, finish_reason, request_id, num_prompt_tokens, max_
tokens_param, req_stats, num_cached_tokens=0):
    # 请求结束时把累计抢占次数写入产出统计，供指标层 observe
    finished_req = FinishedRequestStats(
        finish_reason=finish_reason,
        request_id=request_id,
        num_prompt_tokens=num_prompt_tokens,
        num_preemptions=req_stats.num_preemptions,
        # 其余字段省略
    )
    self.finished_requests.append(finished_req)
```

评论区精华

核心讨论围绕 histogram 的必要性展开：mgoin 质疑 "Why do we need a histogram for this? It doesn't make sense to me beyond reporting the number from the last tick"。作者 linamy85 回应这是 per-request lifecycle 指标，在请求完成时记录，直方图能区分抢占是否均匀分布，还是集中在一小撮长上下文请求上（被抢占 5-10+ 次的 thrashing）；并指出与现有 `vllm:request_*` per-request 指标架构一致。讨论后 simon-mo 批准合并。另有一次设计修正在提交 525335b 中：将 bucket 从包含 0 改为不含 0，因为 Ray metrics 严格要求 histogram 边界为正 (> 0)。

- histogram 语义与必要性 (design): 作者澄清后 simon-mo 批准合并；另在提交 525335b 中修正 bucket 不含 0 以兼容 Ray 指标边界要求。

风险与影响

- 风险:

1. 数据正确性依赖 PREEMPTED 事件流: 若未来多引擎或 DP 场景下事件重复或遗漏, 计数会失真; 当前与节点级计数共用同一事件来源, 行为一致。
2. 兼容性约束: histogram bucket 必须保持正数, 后续扩展 bucket 时需避免回归; 该约束已在提交 525335b 中修复, 但未在测试中显式断言。
3. 测试覆盖有限: 集成测试只断言 series 存在与 count 计数, 未在 CI 中构造真实抢占场景断言分布值, 作者通过外部脚本 simulate_preemption.py 手工验证 (观察到 4 请求共 5 次抢占)。
4. 缺少文档更新: PR 未同步更新 metrics 相关文档, 运维人员需自行理解新指标语义。
 - 影响: 影响范围集中在 v1 指标层: 新增一个 Prometheus 直方图指标, 每请求增加一次整数累加和结束时一次 observe, 性能开销可忽略; 不涉及调度、采样、KV cache 等核心路径。对运维侧, 该指标可用于观察 KV cache 压力下的抢占分布与 thrashing 严重程度, 辅助容量规划与告警策略。对团队而言, 这是请求级指标体系的又一补充, 后续新增指标时需遵循同样的命名与 bucket 正数约束。整体影响程度低、无破坏性变更。
 - 风险标记: 抢占计数依赖 PREEMPTED 事件流完整性, Ray 要求 histogram 边界为正, 测试仅断言 count 未覆盖真实抢占分布, 缺少 Metrics 文档更新

关联脉络

- 暂无明显关联 PR