

# PR #56541 完整报告

vllm-project/vllm

[CI] Sample GPU utilization and memory alongside test timelines

合并时间: 2026-09-12 17:41

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/56541>

## 执行摘要

- 一句话: CI 时间线新增 GPU 利用率与内存采样
- 推荐动作: 值得精读, 尤其适合关注 CI 可观测性和低侵入采集器设计的工程师。重点学习: 如何在零依赖前提下安全采集硬件遥测、如何处理失败而不污染主流程、如何用 spool + OIDC 复用既有认证链路上传数据, 以及字节预算批处理在 OTLP 导出中的必要性。

## 功能与动机

PR body 明确指出目的是“收集 GPU 利用率和已用 / 总内存, 与 CI 命令和 pytest 计时一起, 以便 Builds 时间线区分 GPU-active tests from sampled idle intervals”。同时约束是“不引入 GPU context、pytest-worker sampler、fleet agent 或长期 dashboard token”, 并检查了开放 PR, 确认没有现有实现冲突。这体现了对 CI 可观测性的增强需求, 以及对基础设施最小侵入的追求。

## 实现拆解

1. 新增采样器脚本 `ci_gpu.py`: 定义 `_number`、`parse_samples`、`sample_span`、`collect`、`main`。脚本通过 `nvidia-smi` 的 CSV 输出获取最多 16 个容器可见 NVIDIA 设备的利用率与内存, 解析时过滤非法值 (NaN、负数、超上限) 并跳过 MIG 父设备。`collect` 以 1 秒间隔轮询, 每 30 秒将攒批事件封装为 `ci.gpu.samples span` 并通过 `export_spans` 上传; 最后一小批在命令结束时通过 `record_spans` 落盘 `spool`。
2. 扩展 OTLP 编码与导出 (`ci_otel.py`): `Span dataclass` 新增 `events` 字段, `_encode_span` 将事件按 OTLP span event 格式编码。`export_spans` 从按固定条数切分改为同时受 `MAX_BATCH_SIZE` 和新增的 `MAX_BATCH_BYTES` (1 MB) 约束, 避免含大量采样事件时单请求过大。
3. Shell 生命周期管理 (`ci_otel.sh`): `ci_otel_start` 在满足 `CI_INFRA_GPU_SAMPLING != 0`、`ci_gpu.py` 存在且 `nvidia-smi` 可用时后台启动采样器; `ci_otel_finish` 先发送 `SIGTERM`, 循环等待最多 3 秒, 再强制 `SIGKILL` 并回收进程, 同时保留原命令退出状态。
4. 测试配套 (`tests/test_ci_otel.py`): 新增 6 组 GPU 相关测试, 覆盖解析零值与非法值、事件 `spool` 往返、查询失败有界且不伪造 0%、上传失败丢弃批次、shell 启停与退出状态保留 (含禁用开关参数化)。
5. 文档 (`README.md`): 新增 README 说明采样范围、失败处理、MIG 限制、部署前提与验证方式。

关键文件：

- `.buildkite/scripts/ci-otel/ci_gpu.py`（模块 GPU 采集；类别 source；类型 core-logic；符号 `_number`, `parse_samples`, `sample_span`, `collect`）：新增的 GPU 采样器，负责轮询 `nvidia-smi`、解析 CSV、生成 OTLP 事件并管理批次上传，是本次功能的核心实现。
- `.buildkite/scripts/ci-otel/ci_otel.py`（模块 OTLP 导出；类别 source；类型 core-logic；符号 `Span`, `_encode_span`, `export_spans`）：扩展 `Span` 支持 OTLP events，并为 `export_spans` 引入字节预算批处理，避免携带大量采样事件时请求过大。
- `.buildkite/scripts/ci-otel/tests/test_ci_otel.py`（模块 CI 测试；类别 test；类型 test-coverage；符号 `test_export_batches_with_one_oidc_token`, `test_gpu_samples_preserve_zero_and_omit_unsupported_metrics`, `test_gpu_batches_keep_command_identity_and_survive_spool_roundtrip`, `test_gpu_query_failures_are_bounded_and_never_synthesize_idle_samples`）：新增 GPU 采样相关测试，覆盖解析、spool 往返、查询失败、上传失败和 shell 生命周期，是功能可靠性的主要保障。
- `.buildkite/scripts/ci-otel/ci_otel.sh`（模块 生命周期；类别 other；类型 core-logic）：负责采样器进程的启动与回收，确保不干扰命令退出状态，是采样器可靠运行的生命周期支撑。
- `.buildkite/scripts/ci-otel/README.md`（模块 CI 文档；类别 docs；类型 documentation）：说明了采样器的行为约束、覆盖范围、部署前提和验证方式，是团队理解与部署该功能的入口文档。

关键符号：`collect`, `parse_samples`, `sample_span`, `_number`, `export_spans`, `_encode_span`

## 关键源码片段

### `.buildkite/scripts/ci-otel/ci_gpu.py`

新增的 GPU 采样器，负责轮询 `nvidia-smi`、解析 CSV、生成 OTLP 事件并管理批次上传，是本次功能的核心实现。

```
def collect(parent_pid: int, stop: threading.Event) -> None:
    """按 1 秒间隔轮询 nvidia-smi，攒批后上传 OTLP 事件。"""
    events: list[dict] = []
    batch_started = time.monotonic()
    failures = 0
    try:
        # 父进程退出后（getppid 变化）或收到 stop 信号时停止，避免孤儿进程。
        while not stop.is_set() and os.getppid() == parent_pid:
            started = time.monotonic()
            try:
                result = subprocess.run(
                    [
                        "nvidia-smi",
                        f"--query-gpu={QUERY}",
                        "--format=csv,noheader,nounits",
                    ],
                    capture_output=True,
                    text=True,
```

```

        timeout=QUERY_TIMEOUT_SECONDS,
        check=True,
    )
    if stop.is_set():
        break
    samples = parse_samples(result.stdout, time.time_ns())
    if not samples:
        break
    events.extend(samples)
    failures = 0
except (OSError, subprocess.SubprocessError):
    failures += 1
    # 连续 3 次查询失败即放弃，避免无限重试拖慢命令。
    if failures >= 3:
        print(
            "CI GPU sampling stopped: device query unavailable",
            file=sys.stderr,
        )
        break
if events and time.monotonic() - batch_started >= BATCH_SECONDS:
    # 每 30 秒上传一次；上传失败则丢弃本批，时间线留缺口，
    # 绝不回填 0 利用率，防止误导。
    export_spans([sample_span(events)], timeout_seconds=2)
    events = []
    batch_started = time.monotonic()
    stop.wait(max(0, INTERVAL_SECONDS - (time.monotonic() - started)))
finally:
    # 命令结束时 shell 会先 join 本进程，所以最后一小批通过
    # record_spans 写入 spool，由 job-end 统一上传。
    if events:
        record_spans([sample_span(events)])

```

### [.buildkite/scripts/ci-otel/ci\\_otel.py](#)

扩展 Span 支持 OTLP events，并为 export\_spans 引入字节预算批处理，避免携带大量采样事件时请求过大。

```

def export_spans(spans: list[Span], timeout_seconds: float = 3.0) -> bool:
    """将 spans 编码为 OTLP protobuf 并上传，失败时静默返回 False。"""
    if not spans or os.getenv("BUILDKITE") != "true":
        return False
    try:
        deadline = time.monotonic() + max(timeout_seconds, 0.1)
        token = _oidc_token(deadline)
        batches: list[list[Span]] = []
        batch: list[Span] = []
        size = 0
        for span in spans:
            span_size = len(_encode_span(span))
            # 按字节预算和条数上限双重切分，防止含大量采样事件时

```

```

# 单请求超过网关限制；一个 token 覆盖所有批次。
if batch and (
    size + span_size > MAX_BATCH_BYTES or len(batch) >= MAX_BATCH_SIZE
):
    batches.append(batch)
    batch, size = [], 0
    batch.append(span)
    size += span_size
if batch:
    batches.append(batch)
for batch in batches:
    request = urllib.request.Request(
        ENDPOINT,
        data=encode_request(batch),
        method="POST",
        headers={
            "Authorization": f"Bearer {token}",
            "Content-Type": "application/x-protobuf",
            "User-Agent": "vllm-ci-otel/1",
        },
    )
    with urllib.request.urlopen(
        request, timeout=_remaining_seconds(deadline)
    ) as response:
        if response.status != 200:
            raise RuntimeError(f"OTLP endpoint returned {response.status}")
    return True
except Exception as error:
    print(f"CI timing upload skipped: {error}", file=sys.stderr)
    return False

```

## 评论区精华

本 PR 无实质技术 review 讨论。claude[bot] 仅自动提示仓库配置了手动 code review，未触发技术评审。PR 由维护者直接合并，设计决策主要体现在 PR body 与实现中，例如：采样间隔与批量阈值、查询失败三次即停、上传失败丢弃批次而非无限缓存、MIG父设备指标省略等。

- Claude Bot 自动评论 (other): 无实质技术讨论，PR 由维护者直接合并。

## 风险与影响

- 风险：
  1. 采样器与 CI 命令共生：每个 traced command 会启动一个后台 Python 进程，虽然 canary 显示计时影响很小（12.697–13.152 s vs 12.491–13.332 s），但并未建立精确上界，且未验证 L4 以外的 GPU 类型。
  2. 数据缺口：若周期上传失败，批次会被丢弃，时间线对应区间显示为缺口而不是 0%；短测试（如 475 ms）可能完全没有样本。这在 README 中已明确，但用户可能误读为 GPU 空闲。

3. 覆盖范围限制：仅采集 NVIDIA 命令，AMD/CPU 作业无数据；MIG 父设备指标被省略，对于 MIG 分区的作业无法展示其实际分配的内存。
4. nvidia-smi 依赖：ci\_gpu.py 对 nvidia-smi 的可用性有隐式依赖，连续 3 次查询失败即静默停止；如果节点设备状态异常，可能产生不完整时间线。
5. 字节预算变更：export\_spans 现在每次调用都会调用 \_encode\_span 计算大小，对已有大 span 列表会增加少量 CPU 开销，但换取的是避免超出网关限制。- 影响：对用户（CI 开发者）而言，Builds 时间线将能直看到每个命令 / 测试期间的 GPU 利用率和内存占用，有助于排查性能瓶颈和资源浪费。对系统而言，每个 traced command 新增一个轻量子进程，无 CUDA context、无 PyTorch 导入，开销可控且可开关。对团队而言，需先部署配套的 vllm-dashboard PR 并保留七日样本后才能启用广泛收集；本次改动仅影响受信 CI 追踪中的 NVIDIA 命令，不触碰模型推理路径。- 风险标记：采样器与 CI 命令共生，失败静默降级，上传失败丢弃批次造成时间线缺口，仅覆盖 NVIDIA 命令，AMD/CPU 无采样，MIG 父设备指标缺失

## 关联脉络

- PR #56329 [CI] Give every Buildkite step an explicit key, and a hook to enforce it: 同为 CI 基础设施维护，为 Buildkite 配置增加约束与可观测性，与本 PR 共同提升 CI 流程的可靠性与可诊断性。
- PR #56596 [CI/Build] Give the Torch ABI audit time to start: 同样调整 CI 执行时序与超时策略，但本 PR 聚焦 GPU 采样，二者无直接代码依赖，属于相同维护方向。