PR #45958 完整报告

vllm-project/vllm

[KV Offloading] Add basic offloading metrics

合并时间：2026-07-07 14:26

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/45958

执行摘要

- 一句话：KV 卸载添加基础监控指标：延迟、分配失败、分配大小
- 推荐动作：建议精读该 PR，特别是 scheduler.py 中异步延迟的埋点模式和 metrics.py 中 bucket 的设计。设计模式可复用于 vllm 其他子系统的指标添加。测试代码也值得参考，尤其是使用 request_runner fixture 对 scheduler 行为进行集成验证的方式。

功能与动机

为 KV offloading 提供可观测性，使运维人员能够监控 offload 查找延迟、分配失败和分配大小，以便调优容量和性能。

实现拆解

1. 指标定义：在 vllm/distributed/.../offloading/metrics.py 中新增 _ConnectorMetricName 类，定义三个 connector 侧指标名；在 get_connector_metric_definitions() 中添加对应的 OffloadingHistogramMetadata 和 OffloadingCounterMetadata，并设定合适的 bucket 分桶。
2. scheduler 侧埋点：在 scheduler.py 中引入 time.monotonic()，在 get_num_new_matched_tokens() 的 _lookup 调用前后记录同步延迟并 observe 到 _connector_stats；在 _build_store_jobs 中当 prepare_store 返回 None 时增加分配失败计数器；新增 _maybe_observe_lookup_async_delay 方法，在异步 lookup 第一次返回 None 时记录起始时间，在后续 resolve 或请求结束时观察异步延迟。
3. CPU manager 侧埋点：在 cpu/manager.py 的 prepare_store 中记录每次分配的 block 数量到 allocation_sizes_in_current_batch，并在 get_stats() 中 observe 到 CPU_ALLOCATION_SIZE 直方图。
4. 默认 counter 值：将 increase_counter 的默认增量参数设为 1，简化调用。
5. 测试配套：在 test_scheduler.py 新增 3 个测试用例验证 allocation_failure、lookup_sync_delay、lookup_async_delay 的计数与求和；在 test_manager.py 新增 3 个测试用例验证 CPU allocation size 直方图，包括正常、分配失败和驱逐失败路径；在 test_factory.py 新增验证 CPU_ALLOCATION_SIZE 直方图定义；在 test_metrics.py 新增验证 connector metric 的 bucket 值。
6. 测试工具更新：在 offloading_connector/utils.py 中新增 _record_kv_connector_stats 辅助函数，确保测试 runner 能正确收集 stats 输出。

关键文件：

- vllm/distributed/kv_transfer/kv_connector/v1/offloading/scheduler.py（模块 调度器；类别 source；类型 core-logic；符号 _maybe_observe_lookup_async_delay, deferred_lookup_start_time）：核心变更文件：新增同步 / 异步 lookup 延迟埋点、分配失败计数器、_maybe_observe_lookup_async_delay 方法，以及 deferred_lookup_start_time 字段。
- vllm/distributed/kv_transfer/kv_connector/v1/offloading/metrics.py（模块 指标定义；类别 source；类型 core-logic；符号 _ConnectorMetricName）：新增 _ConnectorMetricName 类定义指标名，并添加到 get_connector_metric_definitions() 中，包含详细 bucket 配置。同时修改 increase_counter 默认参数。
- tests/v1/kv_connector/unit/offloading_connector/test_scheduler.py（模块 测试；类别 test；类型 test-coverage；符号 _reduce_kv_connector_stats, test_scheduler_reports_allocation_failure, test_scheduler_reports_lookup_sync_delay, test_scheduler_reports_lookup_async_delay_on_resolve）：新增 3 个测试用例，验证 scheduler 侧三个指标的记录和 reduce 正确性，引入 _reduce_kv_connector_stats 辅助函数。
- tests/v1/kv_offload/cpu/test_manager.py（模块 测试；类别 test；类型 test-coverage；符号 test_cpu_manager_reports_allocation_size_histogram, test_cpu_manager_reports_allocation_size_on_allocation_failure, fail_allocate_blocks, test_cpu_manager_reports_allocation_size_on_eviction_failure）：新增 3 个测试用例，验证 CPU allocation size 直方图在正常、分配失败和驱逐失败时的行为。
- tests/v1/kv_offload/test_factory.py（模块 测试；类别 test；类型 test-coverage；符号 test_build_metric_definitions_allocation_size_histogram）：新增对 CPU_ALLOCATION_SIZE 直方图定义和 bucket 的验证测试，确保 factory 正确构建 metric 定义。
- tests/v1/kv_connector/unit/offloading_connector/test_metrics.py（模块 测试；类别 test；类型 test-coverage；符号 test_connector_metric_histogram_buckets）：新增 test_connector_metric_histogram_buckets 验证 connector 侧两个查找延迟直方图的 bucket 配置。
- vllm/v1/kv_offload/cpu/manager.py（模块 CPU 管理；类别 source；类型 core-logic）：在 prepare_store 中记录分配大小，在 get_stats 中 observe 直方图。修改 get_stats 返回类型确保非空。
- vllm/v1/kv_offload/cpu/spec.py（模块 配置；类别 source；类型 core-logic）：在 build_metric_definitions 中注册 CPU_ALLOCATION_SIZE 直方图的 metadata。
- vllm/v1/kv_offload/cpu/common.py（模块 常量；类别 source；类型 core-logic）：在 CPUOffloadingMetrics 类中新增 CPU_ALLOCATION_SIZE 常量。
- tests/v1/kv_connector/unit/offloading_connector/utils.py（模块 测试；类别 test；类型 test-coverage；符号 _record_kv_connector_stats）：增强测试工具，确保 runner 能正确捕获并存储 kv_connector_stats。

关键符号：_maybe_observe_lookup_async_delay, get_num_new_matched_tokens, _build_store_jobs, get_connector_metric_definitions, prepare_store, get_stats, increase_counter

关键源码片段

vllm/distributed/kv_transfer/kv_connector/v1/offloading/scheduler.py

核心变更文件：新增同步 / 异步 lookup 延迟埋点、分配失败计数器、_maybe_observe_lookup_async_delay 方法，以及 deferred_lookup_start_time 字段。

```
# vllm/distributed/kv_transfer/kv_connector/v1/offloading/scheduler.py

# 在 RequestOffloadState 中新增字段，用于跟踪异步延迟起始时间
@dataclass(slots=True)
class RequestOffloadState:
    # ... 其他字段 ...
    # time.monotonic() of this request's first deferred offload lookup;
    # None once consumed (observed) or while no lookup is pending.
    deferred_lookup_start_time: float | None = None


def _maybe_observe_lookup_async_delay(
    self, req_status: RequestOffloadState
) -> None:
    start_time = req_status.deferred_lookup_start_time
    if start_time is None:
        return
    req_status.deferred_lookup_start_time = None
    self._connector_stats.observe_histogram(
        _ConnectorMetricName.LOOKUP_ASYNC_DELAY,
        time.monotonic() - start_time,
    )

# 在 get_num_new_matched_tokens 中埋点同步延迟和异步延迟触发
lookup_start = time.monotonic()
num_hit_tokens = self._lookup(req_status)
self._connector_stats.observe_histogram(
    _ConnectorMetricName.LOOKUP_SYNC_DELAY,
    time.monotonic() - lookup_start,
)
if num_hit_tokens is None:
    if req_status.deferred_lookup_start_time is None:
        req_status.deferred_lookup_start_time = lookup_start
else:
    self._maybe_observe_lookup_async_delay(req_status)

# 在 _build_store_jobs 中增加分配失败计数器
if store_output is None:
```

```
        self._connector_stats.increase_counter(
            _ConnectorMetricName.ALLOCATION_FAILURE
        )
```

vllm/distributed/kv_transfer/kv_connector/v1/offloading/metrics.py

新增 _ConnectorMetricName 类定义指标名，并添加到 get_connector_metric_definitions() 中，包含详细 bucket 配置。同时修改 increase_counter 默认参数。

```
# vllm/distributed/kv_transfer/kv_connector/v1/offloading/metrics.py

class _ConnectorMetricName:
    """Connector-side metrics emitted by scheduler-side offloading code."""
    LOOKUP_SYNC_DELAY = "vllm:kv_offload_lookup_sync_delay_seconds"
    LOOKUP_ASYNC_DELAY = "vllm:kv_offload_lookup_async_delay_seconds"
    ALLOCATION_FAILURE = "vllm:kv_offload_allocation_failure"

# 在 get_connector_metric_definitions 中添加:
def get_connector_metric_definitions() -> dict[str, OffloadingMetricMetadata]:
    definitions = {
        # 已有 transfer 指标 ...
        _ConnectorMetricName.LOOKUP_SYNC_DELAY: OffloadingHistogramMetadata(
            documentation=(
                "Histogram of the time spent in a single offload lookup call, "
                "in seconds."
            ),
            buckets=(
                0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005,
                0.01, 0.05, 0.1, 0.5, 1,
            ),
        ),
        _ConnectorMetricName.LOOKUP_ASYNC_DELAY: OffloadingHistogramMetadata(
            documentation=(
                "Histogram of time between a request's offload lookup first "
                "deferring and the following lookup resolving, or request "
                "finish, in seconds."
            ),
            buckets=(
                0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05,
                0.1, 0.5, 1, 5, 10,
            ),
        ),
        _ConnectorMetricName.ALLOCATION_FAILURE: OffloadingCounterMetadata(
            documentation=(
                "Number of KV offload store allocation attempts that failed."
            ),
        ),
    }
    return definitions
```

tests/v1/kv_connector/unit/offloading_connector/test_scheduler.py

新增 3 个测试用例，验证 scheduler 侧三个指标的记录和 reduce 正确性，引入 _reduce_kv_connector_stats 辅助函数。

```
# tests/v1/kv_connector/unit/offloading_connector/test_scheduler.py

def _reduce_kv_connector_stats(runner):
    reduced: dict[str, int | float] = {}
    for payload in runner.kv_connector_stats:
        stats = (
            payload
            if hasattr(payload, "reduce")
            else OffloadingConnectorStats(data=payload)
        )
        for key, value in stats.reduce().items():
            reduced[key] = reduced.get(key, 0) + value
    return reduced

def test_scheduler_reports_allocation_failure(request_runner):
    runner = request_runner(block_size=4, num_gpu_blocks=10, async_scheduling=False)
    runner.new_request(token_ids=[0] * 4)
    # 让 prepare_store 返回 None 模拟分配失败
    runner.manager.prepare_store.side_effect = lambda keys, req_context: None
    runner.run(decoded_tokens=[EOS_TOKEN_ID])
    reduced = _reduce_kv_connector_stats(runner)
    assert reduced[_ConnectorMetricName.ALLOCATION_FAILURE] == 1

def test_scheduler_reports_lookup_sync_delay(request_runner):
    runner = request_runner(block_size=4, num_gpu_blocks=10, async_scheduling=False)
    runner.new_request(token_ids=[1] * 4)
    runner.manager.prepare_store.side_effect = lambda keys, req_context: (
        generate_store_output([])
    )
    runner.run(decoded_tokens=[EOS_TOKEN_ID])
    reduced = _reduce_kv_connector_stats(runner)
    assert reduced[f"{_ConnectorMetricName.LOOKUP_SYNC_DELAY}_count"] == 1
    assert reduced[f"{_ConnectorMetricName.LOOKUP_SYNC_DELAY}_sum"] > 0
```

评论区精华

- 审核者 orozery 建议将单一的 lookup delay 拆分为同步 / 异步两个独立指标，并分别设定不同的 bucket 范围（sync: 0.00001~1s，async: 0.0001~10s）。
- 在 RequestOffloadState 中新增 deferred_lookup_start_time 字段代替全局 dict，用于跟踪异步延迟起始点。
- orozery 要求将 increase_counter 的默认增量值设为 1，简化调用。
- 对于 CPU 分配大小，orozery 指出不应观察空分配（len(keys_to_store)==0），并应在 eviction 之前记录分配大小，以便捕获 eviction 失败时的值。

- 代码中 self._connector_stats 的 lazy 初始化被改为始终非空，并通过 is_empty() 判断是否丢弃空 stats。
- 将 lookup delay 拆分为同步 / 异步两个指标 (design): 接受建议，实现两个 metrics: LOOKUP_SYNC_DELAY 和 LOOKUP_ASYNC_DELAY。
- 使用 RequestOffloadState 字段代替全局 dict 跟踪异步延迟 (design): 采用该方案，新增 deferred_lookup_start_time 字段，并在 consume 后置 None。
- defualt increase_counter 增量值设为 1 (design): 修改方法签名，默认增量为 1。
- CPU allocation size 应在 eviction 之前记录 (design): 将 allocation_sizes_in_current_batch.append(len(keys_to_store)) 移到 eviction 逻辑之前。
- 历史 bucket 值微调 (design): 接受建议，最终 bucket 列表包含 0.00001、0.00005 等。

风险与影响

- 风险：主要风险在 scheduler 的 hot path 中新增 time.monotonic() 调用，但该调用开销极低（纳秒级），对性能影响可忽略。stats 对象的生命周期通过 is_empty() 守卫，不会在无观测时产生不必要的传输。CPU manager 中 allocation_sizes_in_current_batch 列表在每次 get_stats() 后清空，但没有保护防止并发访问（当前为单线程模型，安全）。测试充分覆盖了正常 / 失败 / 驱逐路径，降低回归风险。
- 影响：用户 / 运维：新 metric 默认启用，通过 Prometheus 或 stats logger 暴露，运维可直接监控 KV offloading 延迟和分配失败，无需额外配置。系统：增加少量内存用于观测数据，但每次 get_stats() 后即释放，内存占用可控。团队：该 PR 为后续更精细的 offload 监控（如按 tier 细分）打下基础。影响范围限定在 kv-offloading 子系统，不影响其他核心路径。
- 风险标记：hot-path 增加 time.monotonic() 调用，stats 对象空值处理，测试覆盖充分

关联脉络

- PR #46544 [kv_offload] Establish tier-owned KV event handling: 同一 KV offloading 子系统的基础设施重构，为本 PR 的指标埋点提供了稳定的事件和 stats 框架基础。
- PR #47274 [KV Offload] Add ParentManager ABC for secondary tier callbacks: 引入的父类 ABC 与 CPU manager 中的 allocation size 记录间接相关，确保了二级 tier 回调的一致性。
- PR #46972 [Bugfix][KV offload] Store interior chunk-boundary blocks under MTP/Eagle: 修复了调度器中的 chunk 存储 bug，本 PR 的 allocation_failure 指标可以帮助检测此类问题。