PR #45737 完整报告

vllm-project/vllm

[KV-Offloading] : Expose CPU cache usage metric

合并时间：2026-06-21 05:21

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/45737

执行摘要

- 一句话：暴露 CPU KV 卸载缓存使用百分比指标
- 推荐动作：值得精读，尤其是 get_stats 中的计算逻辑与阈值联动方式。设计讨论也值得关注：指标定义应该由 Spec 集中管理还是允许 Manager 自主注册？该 PR 选择了前者（通过 Spec 的 build_metric_definitions），并通过 CPUOffloadingMetrics 集中常量，是一个合理折衷。

功能与动机

PR 正文明确说明目的："Add vllm:kv_offload_cpu_cache_usage_perc metric to export CPU cache usage percent. The naming and the semantics is designed to match the gpu counter part vllm:kv_cache_usage_perc"。讨论中也提到需要监控 CPU 缓存容量是否饱和，以便运维决策。

实现拆解

1. 定义指标常量集中类：在 vllm/v1/kv_offload/cpu/common.py 中新增 CPUOffloadingMetrics 类，统一存放两个指标名称常量 STORES_SKIPPED 和 CPU_CACHE_USAGE_PERC，替换原有分散的模块级常量。
2. 注册指标定义：修改 vllm/v1/kv_offload/cpu/spec.py 中的 CPUOffloadingSpec.build_metric_definitions，使其总是返回 CPU_CACHE_USAGE_PERC 的 OffloadingGaugeMetadata（无论 store_threshold 如何），而对于 STORES_SKIPPED 仍仅当 store_threshold >= 2 时添加 OffloadingCounterMetadata。
3. 实现指标计算与暴露：修改 vllm/v1/kv_offload/cpu/manager.py 的 get_stats 方法：移除了之前 store_threshold < 2 时直接返回 None 的早期退出；现在始终创建 OffloadingConnectorStats，通过 num_used = _num_allocated_blocks - len(_free_list) - _num_evictable_cache_blocks 计算已用块数，再除以总块数得到使用率（避免除零），设置 gauge；同时仅当 store_threshold >= 2 时执行计数器累加。
4. 编写单元测试：在 tests/v1/kv_offload/cpu/test_manager.py 中添加 test_cpu_manager_reports_cache_usage_gauge 函数，通过 make_cpu_manager 构造不同状态的 manager 并调用 get_stats，验证使用率返回正确值（包括零容量、空、半满、全满、complete_store 释放等场景）。

关键文件：

- vllm/v1/kv_offload/cpu/common.py（模块 指标定义；类别 source；类型 core-logic；符号 CPUOffloadingMetrics）：定义 CPUOffloadingMetrics 类，集中管理所有 CPU 卸载指标的常量名称，是整个指标的源头。
- vllm/v1/kv_offload/cpu/spec.py（模块 注册入口；类别 source；类型 dependency-wiring；符号 CPUOffloadingSpec.build_metric_definitions）：CPUOffloadingSpec.build_metric_definitions 注册新 gauge 指标，并根据 store_threshold 条件注册 counter，是指标正式暴露的入口。
- vllm/v1/kv_offload/cpu/manager.py（模块 卸载管理；类别 source；类型 dependency-wiring；符号 CPUOffloadingManager.get_stats）：CPUOffloadingManager.get_stats 实现使用率计算，将内部状态转换为 Prometheus gauge 值，是核心计算逻辑所在。
- 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_cache_usage_gauge, check_usage_stats）：新增的 test_cpu_manager_reports_cache_usage_gauge 和辅助函数 check_usage_stats 覆盖了各种缓存状态下的使用率计算，确保指标正确性。

关键符号：CPUOffloadingMetrics, CPUOffloadingSpec.build_metric_definitions, CPUOffloadingManager.get_stats, test_cpu_manager_reports_cache_usage_gauge, check_usage_stats

关键源码片段

vllm/v1/kv_offload/cpu/common.py

定义CPUOffloadingMetrics类，集中管理所有CPU卸载指标的常量名称，是整个指标的源头。

```
# vllm/v1/kv_offload/cpu/common.py
# CPUOffloadingMetrics 将 CPU 卸载相关的所有 Prometheus 指标名称集中在一个命名空间类下，
# 避免模块级常量分散，同时方便后续增加新指标。
class CPUOffloadingMetrics:
    # 复用已有的 stores_skipped 计数器名称（不变）
    STORES_SKIPPED = "vllm:kv_offload_stores_skipped"
    # 新引入的 CPU 缓存使用百分比 gauge，命名与 GPU 侧 `vllm:kv_cache_usage_perc` 对齐
    CPU_CACHE_USAGE_PERC = "vllm:kv_offload_cpu_cache_usage_perc"


class CPULoadStoreSpec(BlockIDsLoadStoreSpec):
    """... unchanged ..."""
```

tests/v1/kv_offload/cpu/test_manager.py

新增的 test_cpu_manager_reports_cache_usage_gauge 和辅助函数 check_usage_stats 覆盖了各种缓存状态下的使用率计算，确保指标正确性。

```
# tests/v1/kv_offload/cpu/test_manager.py
# 测试 CPU 缓存使用率 gauge 的准确性

def test_cpu_manager_reports_cache_usage_gauge():
```

```
    # 辅助检查函数：获取 stats 并断言 gauge 值
    def check_usage_stats(manager: CPUOffloadingManager, value: float):
        stats = manager.get_stats()
        assert stats is not None
        # 使用 approx 处理浮点
        assert stats.reduce()[CPUOffloadingMetrics.CPU_CACHE_USAGE_PERC] == pytest.
        approx(value)

    # 场景 1：零容量管理器始终返回 0.0
    manager = make_cpu_manager(num_blocks=0)
    check_usage_stats(manager, 0.0)

    # 场景 2：空管理器（4 块，未分配）返回 0.0
    manager = make_cpu_manager(num_blocks=4)
    check_usage_stats(manager, 0.0)

    # 场景 3：分配 2/4 块后返回 0.5
    manager.prepare_store(to_keys([1, 2]), _EMPTY_REQ_CTX)
    check_usage_stats(manager, 0.5)

    # 场景 4：填满 4 块后返回 1.0
    manager.prepare_store(to_keys([3, 4]), _EMPTY_REQ_CTX)
    check_usage_stats(manager, 1.0)

    # 场景 5：complete_store 后块变为可驱逐，使用率下降
    manager.complete_store(to_keys([1, 2]), _EMPTY_REQ_CTX)
    check_usage_stats(manager, 0.5)

    manager.complete_store(to_keys([3, 4]), _EMPTY_REQ_CTX)
    check_usage_stats(manager, 0.0)
```

评论区精华

指标定义归属争议：orozery 最初质疑在 base.py 增加默认 build_metric_definitions 是否必要（已有 OffloadingSpec 的同名方法），varun 解释这是为了让 Manager 能注册自己特有的指标，但最终被要求移回 cpu/spec.py 通过基类机制注册。

指标名称与文档修订：orozery 建议将指标名定为 vllm:kv_offload_cpu_cache_usage_perc 以匹配 GPU 侧 vllm:kv_cache_usage_perc 的命名惯例，ronensc 表示同意。在文档字符串上，orozery 建议更精确的描述，varun 最初指出 get_stats 中计算包含非活动块会导致指标始终为 1.0，随后 orozery 承认这一点并明确应该只计数“当前已被钉住（pinned）”的块（即活跃传输中的块）。varun 随后将计算调整为扣除 _num_evictable_cache_blocks，使指标反映活跃请求占用的比例，并更新了文档。

测试归宿：orozery 反对将 CPU 指标测试放在通用 test_metrics.py 中，认为它属于 CPU 侧测试文件，最终测试代码被放置在 test_manager.py。

- build_metric_definitions 的归属：Spec 还是 Manager？ (design): 指标定义仍然通过 Spec 的 build_metric_definitions 注册，Manager 只负责计算和输出；常量则集中到

CPUOffloadingMetrics 类中。

- 使用率计算语义：是否包含非活动块？ (correctness): 使用率 = （已分配块 - 空闲块 - 可驱逐缓存块）/ 总块数， 只反映当前被活跃传输占用的比例。
- 指标名称与文档字符串的迭代 (design): 最终指标名称为 vllm:kv_offload_cpu_cache_usage_perc，文档字符串清晰描述了含义和饱和条件。
- 测试位置与方式 (testing): 测试只保留在 tests/v1/kv_offload/cpu/test_manager.py 中。

风险与影响

- 风险：
  1. 行为变化：get_stats 现在始终返回 OffloadingConnectorStats 对象，而以前在 store_threshold < 2 时返回 None。调用方需确保不会因为空值检查而错过统计采集。审阅中未发现显式依赖 None 的消费者，但需关注是否有自定义监控脚本依赖旧行为。
  2. 计算准确性：使用率计算依赖于 _num_evictable_cache_blocks 的维护正确性。如果该字段在 eviction 或 cache reset 时未及时更新，会导致指标偏差。
  3. 指标注册：新指标通过 spec.py 的 build_metric_definitions 注册，若其他 Spec 也注册同名指标可能导致冲突。目前仅 CPUOffloadingSpec 注册该 gauge，风险可控。 - 影响：影响范围限于 CPU KV 卸载子系统（vllm/v1/kv_offload/cpu/）。用户可通过 Prometheus 查询 vllm:kv_offload_cpu_cache_usage_perc 指标，用于监控 CPU 缓存饱和度和运维告警。团队需要确保指标名称与 GPU 侧一致，并对后续可能添加的其他卸载后端（如 NPU）给出命名模板。 - 风险标记：get_stats 返回值行为变化，新指标计算依赖 _num_evictable_cache_blocks 正确性，指标命名需与 GPU 侧协调

关联脉络

- PR #45757 现有讨论中提到的后续 PR，用于完善可驱逐块统计：讨论中 varun 提到需要先合并 #45757 以使用 _num_idle_cache_blocks 来更准确地计算使用率。但该 PR 尚未合并，因此当前实现使用了 _num_evictable_cache_blocks。