PR #42206 完整报告

vllm-project/vllm

[Metrics] Add group-aware KV cache capacity to vllm:cache_config_info

合并时间：2026-06-12 19:49

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/42206

执行摘要

- 一句话：为 vllm:cache_config_info 添加 group-aware KV cache 容量指标
- 推荐动作：值得精读，特别是以下设计决策：
  - 如何将运行时计算的状态从 engine core 通过 ReadyResponse 同步到前端。
  - compute_hash 忽略字段的处理确保新增字段不影响编译图。
  - per-engine 与总和的取舍体现了对 DP 架构的正确建模。

功能与动机

Addresses the Prometheus vs. startup-log discrepancy for KV cache capacity in #42024. The startup log already reports the correct group-aware KV cache capacity for hybrid models, but Prometheus did not expose matching info in vllm:cache_config_info.

实现拆解

1. 重构容量计算：将 _report_kv_cache_config 改为公开函数 get_kv_cache_capacity，返回 (num_tokens, max_concurrency) 元组；将日志调用移入 get_kv_cache_configs 中，避免重复。
2. CacheConfig 新增字段：在 CacheConfig 中增加 kv_cache_size_tokens 和 kv_cache_max_concurrency 两个 init=False 的字段，并在 compute_hash 的忽略因子中添加，避免影响编译哈希。
3. engine core 初始化时填充：在 V1Engine._initialize_kv_caches 中分配 KV cache 后，调用 get_kv_cache_capacity 并写入 cache_config，随后在 process_input_sockets 中作为 EngineCoreReadyResponse 的一部分发送给前端。
4. 前端同步：在 MPClient._apply_ready_response 中从 response 同步两个字段到本地 cache_config。采用先有值不覆盖的策略，且不跨 DP 累加（保持 per-engine 语义）。
5. 测试配套：在 tests/entrypoints/serve/instrumentator/test_metrics.py 中增加对 vllm:cache_config_info 标签值的断言，验证两个字段不为空。同时微调 test_kv_cache_utils.py 适应重构。

关键文件：

- vllm/v1/core/kv_cache_utils.py（模块 缓存工具；类别 source；类型 core-logic；符号 _report_kv_cache_config, get_kv_cache_capacity）：核心计算逻辑：新增 get_kv_cache_capacity 函数，替代 _report_kv_cache_config；在 get_kv_cache_configs 中内联日志调用。
- vllm/v1/engine/core_client.py（模块 客户端；类别 source；类型 core-logic）：前端同步关键：在 _apply_ready_response 中从 response 读取并写入 cache_config，采用 per-engine 语义。
- vllm/v1/engine/core.py（模块 引擎核心；类别 source；类型 core-logic）：engine core 初始化入口：在 _initialize_kv_caches 中计算并存储容量；在 process_input_sockets 中打包到 ReadyResponse。
- vllm/config/cache.py（模块 配置层；类别 source；类型 core-logic）：数据模型扩展：CacheConfig 新增 kv_cache_size_tokens 和 kv_cache_max_concurrency 字段，并加入 compute_hash 忽略列表。
- vllm/v1/engine/__init__.py（模块 协议定义；类别 source；类型 core-logic）：通信协议扩展：EngineCoreReadyResponse 新增两个可选字段用于传输容量信息。
- tests/entrypoints/serve/instrumentator/test_metrics.py（模块 指标测试；类别 test；类型 test-coverage）：集成测试：验证 vllm:cache_config_info 中新增标签存在且非空。
- tests/v1/core/test_kv_cache_utils.py（模块 缓存测试；类别 test；类型 test-coverage）：单元测试适配：微调测试以兼容重构后的函数签名。

关键符号：get_kv_cache_capacity, _apply_ready_response, _initialize_kv_caches, process_input_sockets, compute_hash

关键源码片段

vllm/v1/core/kv_cache_utils.py

核心计算逻辑：新增 get_kv_cache_capacity 函数，替代 _report_kv_cache_config；在 get_kv_cache_configs 中内联日志调用。

```
# 来源文件：vllm/v1/core/kv_cache_utils.py

# --- 核心计算函数：获取 group-aware KV cache 容量 ---
def get_kv_cache_capacity(
    vllm_config: VllmConfig, kv_cache_config: KVCacheConfig
) -> tuple[int, float]:
    """
    Get the group-aware KV cache token capacity and max concurrency.
    """
    max_model_len = vllm_config.model_config.max_model_len
    # 从 group-aware 并发度计算 tokens 容量
    max_concurrency = get_max_concurrency_for_kv_cache_config(
        vllm_config, kv_cache_config
    )
    # 容量（单位：tokens）= max_concurrency * max_model_len
    return int(max_concurrency * max_model_len), max_concurrency
```

```
# --- 在 get_kv_cache_configs 中替代旧 _report_kv_cache_config 调用 ---
if len(kv_cache_config.kv_cache_groups) > 0:
    max_model_len = vllm_config.model_config.max_model_len
    num_tokens, max_concurrency = get_kv_cache_capacity(
        vllm_config, kv_cache_config
    )
    logger.info_once("GPU KV cache size: %s tokens", f"{num_tokens:,}")
    logger.info_once(
        "Maximum concurrency for %s tokens per request: %.2fx",
        f"{max_model_len:,}",
        max_concurrency,
    )
```

vllm/v1/engine/core_client.py

前端同步关键：在 _apply_ready_response 中从 response 读取并写入 cache_config，采用 per-engine 语义。

```
# 来源文件：vllm/v1/engine/core_client.py

# --- 多点进程前端中同步 engine core 返回的 KV cache 容量 ---
def _apply_ready_response(self, payload: bytes) -> None:
    # ... 前置代码处理 num_gpu_blocks, block_size 同步 ...
    cache_config = vllm_config.cache_config
    cache_config.block_size = response.block_size

    # 不跨 DP 累加，保持 per-engine 值
    cache_config.kv_cache_size_tokens = (
        getattr(cache_config, "kv_cache_size_tokens", None)
        if getattr(cache_config, "kv_cache_size_tokens", None) is not None
        else response.kv_cache_size_tokens
    )
    cache_config.kv_cache_max_concurrency = (
        getattr(cache_config, "kv_cache_max_concurrency", None)
        if getattr(cache_config, "kv_cache_max_concurrency", None) is not None
        else response.kv_cache_max_concurrency
    )
    # ...
```

评论区精华

主要讨论集中在三个设计决策：

- 不要新增独立 Prometheus gauge：markmc 要求避免增加新指标，而是将字段追加到已有的 vllm:cache_config_info，作者同意并移除独立 gauge。
- 复用现有计算逻辑：markmc 建议重构 _report_kv_cache_config 为公共计算函数，作者实现 get_kv_cache_capacity 并在两处调用。
- 测试文件选择：markmc 认为 AI 生成的完整单元测试过于臃肿，应集成到现有 test_metrics.py 中，作者移除独立测试文件并添加少量断言。 此外，gemini-code-assist

指出 DP 模式下应跨 *engine* 求和，但作者坚持 *per-engine* 语义且 markmc 最终批准，未修改。

- 不要新增独立 Prometheus *gauge*，改为追加到 *cache_config_info* (design): 作者移除独立 *gauge* 并改用 *cache_config_info* 扩展标签。
- 复用 _report_kv_cache_config 计算逻辑 (design): 作者重构为公开函数 *get_kv_cache_capacity*，日志调用移至 *get_kv_cache_configs*。
- 测试应集成到现有 test_metrics.py 而非新建文件 (testing): 作者移除独立测试文件，在现有 test_metrics.py 中添加少量断言。
- DP 模式下应跨 *engine* 求和还是保持 *per-engine* (correctness): 作者坚持 *per-engine* 语义，注释说明不求和的原因，markmc 最终批准，未采纳求和建议。

风险与影响

- 风险:
  1. *per-engine* 值可能误导：在 DP 部署中，两个指标反映单个 *engine* 的容量而非集群总量，用户若将它们理解为全局容量会导致误判。代码注释和 *commit message* 已说明语义，但文档未同步更新。
  2. 仅 v1 路径生效：旧版 v0 *engine* 代码路径未覆盖，仍缺失这两个字段，向前兼容性良好但可能造成混淆。
  3. 核心路径变更：_initialize_kv_caches 和 _apply_ready_response 均在核心初始化 / 同步路径上，任何计算错误可能导致容量报告异常。
  4. 前端兼容性：*EngineCoreReadyResponse* 新增可选字段，旧前端反序列化时需忽略额外字段，msgspec 默认忽略 unknown，风险较低。 - 影响：用户：混合模型用户现在可以通过 Prometheus 直接查看准确的 KV *cache* 容量，无需手动计算。*per-engine* 语义需在文档中明确。系统：无性能影响，仅增加一次同步调用和两个字段存储。团队：为未来更细粒度的 KV *cache* 监控提供基础。
- 风险标记：核心路径变更，*per-engine* 值需谨慎解释，v0 未覆盖

关联脉络

- 暂无明显关联 PR