PR #27174 完整报告

sgl-project/sglang

Add num_waiting_uncached_tokens load metric

合并时间：2026-06-04 09:29

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27174

执行摘要

- 一句话：新增 num_waiting_uncached_tokens 负载指标
- 推荐动作：值得阅读以了解如何在不重复遍历缓存的情况下利用现有关联信息推导派生指标。其中 supports_fast_match_prefix 接口设计为不同缓存后端提供了优雅的扩展点，可作为类似场景的参考模式。

功能与动机

仅靠 num_waiting_reqs 无法区分 50 个短 prompt 和 50 个长 prompt，也无法区分 99% 已缓存和 1% 已缓存的请求。路由器和自动扩缩器需要 token 级别的信号来做出关于 prefill 反压的明智决策。

实现拆解

1. Req 类添加字段：在 schedule_batch.py 的 Req 中新增 num_matched_prefix_tokens 属性，记录请求已匹配到缓存的 token 数（设备端 prefix_indices 加上 host 端命中，受 _compute_max_prefix_len 限制）。在 match_prefix_for_req 函数（schedule_policy.py）中计算并设置该字段。
2. 缓存接口扩展：在 base_prefix_cache.py 的抽象基类 BasePrefixCache 中添加 supports_fast_match_prefix 方法，默认返回 False。子类可重写以指示支持快速前缀匹配，用于决定是否在缓存无关调度策略中执行前缀匹配以收集指标。
3. 调度策略适配：在 SchedulePolicy.calc_priority 中，对非缓存感知策略（如 FCFS），若缓存支持快速匹配且非 decode 模式，则对等待队列中的每个请求调用 match_prefix_for_req 以填充 num_matched_prefix_tokens，确保负载指标可用。
4. 负载查询逻辑：在 load_inquirer.py 的 LoadInquirer 类中新增 get_num_waiting_uncached_tokens 方法，遍历 waiting queue 各请求，累加 seqlen - num_matched_prefix_tokens；同时对 chunked prefill 中的请求扣除 len(prefix_indices)。在 get_loads 中调用并返回该值。
5. 输出与序列化：在 io_struct.py 的 GetLoadsReqOutput 和 load_snapshot.py 的 LoadSnapshot 中添加 num_waiting_uncached_tokens 字段定义与序列化支持。
6. 测试更新：在 test_radix_force_miss.py 的 _StubReq 中添加 _compute_max_prefix_len 和 num_matched_prefix_tokens 以适配新接口。

关键文件：

- python/sglang/srt/managers/scheduler_components/load_inquirer.py（模块 调度器；类别 source；类型 core-logic；符号 get_num_waiting_uncached_tokens）：核心实现文件：新增 get_num_waiting_uncached_tokens 方法计算未缓存 token 数，并在 get_loads 中返回新指标。
- python/sglang/srt/managers/schedule_policy.py（模块 调度策略；类别 source；类型 core-logic；符号 calc_priority, match_prefix_for_req）：调度策略适配：在 match_prefix_for_req 中设置 num_matched_prefix_tokens；在 calc_priority 中为非缓存感知策略补充前缀匹配以填充指标。
- python/sglang/srt/managers/io_struct.py（模块 数据结构；类别 source；类型 data-contract；符号 GetLoadsReqOutput）：数据定义：在 GetLoadsReqOutput 中添加 num_waiting_uncached_tokens 字段及度量元信息。
- python/sglang/srt/mem_cache/base_prefix_cache.py（模块 缓存层；类别 source；类型 core-logic；符号 supports_fast_match_prefix）：抽象接口：新增 supports_fast_match_prefix 基础方法，默认返回 False，子类可重写以启用快速匹配。
- python/sglang/srt/managers/schedule_batch.py（模块 批处理；类别 source；类型 data-contract；符号 Req）：数据结构扩展：在 Req 类中添加 num_matched_prefix_tokens 字段，并在 reset_for_retract 中重置。
- python/sglang/srt/managers/load_snapshot.py（模块 负载快照；类别 source；类型 data-contract；符号 LoadSnapshot）：序列化支持：在 LoadSnapshot 结构体及其 to_dict 方法中添加 num_waiting_uncached_tokens 字段。
- test/registered/unit/mem_cache/test_radix_force_miss.py（模块 测试；类别 test；类型 test-coverage；符号 _compute_max_prefix_len）：测试适配：更新 StubReq 以包含 _compute_max_prefix_len 和 num_matched_prefix_tokens，保持与新 API 兼容。

关键符号：get_num_waiting_uncached_tokens, supports_fast_match_prefix, match_prefix_for_req, calc_priority

关键源码片段

python/sglang/srt/managers/scheduler_components/load_inquirer.py

核心实现文件：新增 get_num_waiting_uncached_tokens 方法计算未缓存 token 数，并在 get_loads 中返回新指标。

```
def get_num_waiting_uncached_tokens(self) -> int:
    """获取等待预填充计算的未缓存输入 token 数量。"""
    # 在 decode 模式下从不进行预填充，因此返回 0
    if self.disaggregation_mode == DisaggregationMode.DECODE:
        return 0
    num_tokens = 0
    # 遍历等待队列中的每个请求
    for req in self.get_waiting_queue():
        # seqlen 是输入 token 总数，num_matched_prefix_tokens 是已被缓存覆盖的部分
        # 两者之差即为仍需 prefill 计算的 token 数
        # 若匹配在等待队列中禁用，则该指标回退为 seqlen
        num_tokens += max(0, req.seqlen - req.num_matched_prefix_tokens)
```

```
    # 处理当前正在 chunked prefill 的请求（部分 token 可能在本次迭代中已完成）
    cr = self.get_chunked_req()
    if cr is not None:
        # 使用 prefix_indices 而非 num_matched_prefix_tokens，因为 chunked req
        的前缀索引已更新
        num_tokens += max(0, cr.seqlen - len(cr.prefix_indices))
    return num_tokens
```

python/sglang/srt/managers/schedule_policy.py

调度策略适配：在 match_prefix_for_req 中设置 num_matched_prefix_tokens；在 calc_priority 中为非缓存感知策略补充前缀匹配以填充指标。

```
def match_prefix_for_req(
    req: Req,
    token_ids: Optional[array[int]] = None,
    *,
    cow_mamba: bool = False,
    include_req: bool = False,
):
    # ... 此前缀匹配逻辑省略 ...
    # 新增：计算并保存已匹配的 token 数
    max_len = req._compute_max_prefix_len(len(token_ids))
    req.num_matched_prefix_tokens = min(
        len(req.prefix_indices) + req.host_hit_length, max_len
    )
    return match_result

# calc_priority 中的新增逻辑
if (
    not isinstance(policy, CacheAwarePolicy)
    and self.tree_cache.supports_fast_match_prefix()
    and get_global_server_args().disaggregation_mode != "decode"
):
    for r in waiting_queue:
        match_prefix_for_req(self.tree_cache, r)
```

python/sglang/srt/managers/io_struct.py

数据定义：在 GetLoadsReqOutput 中添加 num_waiting_uncached_tokens 字段及度量元信息。

```
@dataclass
class GetLoadsReqOutput(BaseReq):
    # ... 其他字段 ...
    num_waiting_uncached_tokens: int = field(
        metadata={
            "metric": (
                "gauge",
                "Number of uncached input tokens waiting for prefill compute",
            )
```

```
        }
    )
    # ... 后续字段 ...
```

评论区精华

唯一的审查讨论聚焦于字段命名：merrymercy 指出原先的 matched_prefix_len 可能被误解为 len(matched_prefix)，建议更具体，避免混淆。cctry 随后将其重命名为 num_matched_prefix_tokens 并补充注释说明语义。

- 字段命名 clarity (style): cctry 将其重命名为 'num_matched_prefix_tokens' 并补充注释，问题已解决。

风险与影响

- 风险：主要风险在于新增的前缀匹配开销：对非缓存感知策略，若 supports_fast_match_prefix 为 true，则每次调度时会对整个 waiting queue 执行一次前缀匹配。但该操作仅发生在配置启用且非 decode 模式下，且匹配逻辑本身已在缓存感知策略中执行，故性能影响有限。其次，num_matched_prefix_tokens 在 reset_for_retract 中已重置为 0，无状态泄漏。API 兼容性方面，新增字段对已有消费方透明，无破坏性变更。
- 影响：用户可通过 /v1/loads 获得更精确的 prefill 负载信号，辅助负载均衡和扩缩容决策。系统额外开销可控。团队需理解新指标含义，但无需修改现有配置。影响范围限于调度与可观测性组件，属中等程度优化。
- 风险标记：前缀匹配开销，字段重置，API 兼容

关联脉络

- 暂无明显关联 PR