

PR #39168 完整报告

sgl-project/sglang

[Router] Add --worker-queue-limit: stop sending cache-affinity traffic to a queueing worker

合并时间: 2026-09-16 13:24

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/39168>

执行摘要

本 PR 为 Router 的 Cache-Aware 路由引入 `--worker-queue-limit`: 当某个 worker 的引擎已经有 N 个（含）以上请求在排队时，它就被移出缓存亲和候选集，请求改投持有同一前缀的其他 worker，否则落到未排队的最轻载 worker。改动跨越准入 (`policies/admission.rs`)、选择阶梯 (`policies/selection.rs`)、指标 (`server/metrics.rs`)、配置面 (`config/cli.rs` / `config/types.rs`) 与数据契约 (`policies/mod.rs`)，共 11 个文件、+1354/-74，并在两轮 review 后修正了两处「指标报告的与文档承诺不一致」的正确性问题。未设置该 flag 时行为完全不变。

功能与动机

Cache-Aware 路由此前按前缀命中加压力 / 容量挑 worker，会把请求送给一个正在让请求排队的引擎。PR body 说明了为什么必须看「等待队列」而不是「运行深度」：

on long-prompt traffic they diverge. An engine can start queueing at 7-8 running, far below its running cap, so a depth threshold either fires on healthy busy workers or misses the workers actually making requests wait.

闸门读的是 `EngineWorkerLoad::num_waiting_reqs`，来自 #34608 的基础负载采样——它直接回答「我会不会先到别人的后面排队」。第二个动机是可观测性：

a fully saturated fleet reads as a healthy one, which is the exact blind spot this series exists to remove.

因此本 PR 同时引入决策标签 `sgl_router_cache_aware_decisions_total` 与分流代价直方图 `sgl_router_diverted_overlap_blocks`，并刻意让 `all_queued` 不冠以 `cache_hit` 前缀，避免被命中率查询 `decision=~"cache_hit.*"` 吸收。

实现拆解

1. 配置面 (`config/types.rs`、`config/cli.rs`、`server/routes/chat.rs`) : `AffinityConfig` 新增 `worker_queue_limit`: `Option<u64>`，默认 `None`，注释解释阈值随 `--dp-size` 放大（样本按 worker 的 DP rank 求和）；`cli.rs` 新增同名 `#[arg(long)]`，并入 `tuned_cache_candidates` 以强制 `--policy cache_aware`，并把 `Some(0)` 的值检查放在策略检查之前（注释：值错误应优先于策略错误）；入口 `chat.rs` 只有 +13 行，把阈值塞进 `PrefillSelectionInputs`——PR body 对比指出，没有 #39322 的 `selection seam` 时这一步要在 `chat.rs` 写 +112/-7。

- 2. 唯一判据 (policies/admission.rs) : queue_gate_admits 编码两个决定——边界用 < (等于 limit 的 worker 已经在让请求等待), 无新鲜样本时 is_none_or 放行 (fail open, 因为 router 侧 in-flight 计数不能区分「在跑」与「在等」); fleet_is_all_queued 要求 limit 已设、fleet 非空、且全 fleet 都有新鲜样本并超限, 避免把「未知」当成「已满」。
- 3. 候选解析: 闸门先于硬准入 (resolve_cache_candidates) : 新增 fleet 参数; 单次遍历把候选拆成 evaluated 与被摘掉集合, 同时累计 queue_gate_rejected_candidates 与 queue_gate_best_rejected_blocks; FreshLoadLookup 改为围绕 evaluated 构建, 防止被摘候选缺 monitor 数据而把压力比较静默降级为 router-local; admission_evaluated_candidates 只统计真正进入硬准入的候选, 因此与闸门拒绝天然不相交。
- 4. 第二层兜底与提案降级: queue_gate_fell_back 只在「闸门清空了候选集 + 全 fleet 饱和」时重新接纳未过闸门的候选; resolve_prefill / resolve_prefill_admitted 新增 queue_limit , 把排队中的 primary/backup 降级到两级 range fallback 而非直接拒绝, 只有 CapacityFallbackPowerOfTwo 最后手段有意不接闸门 (此时域内已无未排队且容量可用的落点)。
- 5. 指标与记账 (server/metrics.rs、policies/selection.rs) : 新增 CacheAwareDecision 四态、sgl_router_cache_aware_decisions_total{model_id,decision}、sgl_router_diverted_overlap_blocks{model_id} (OVERLAP_BLOCK_BUCKETS 为 2 的幂到 8192 块); selection.rs 用纯函数 cache_aware_fallback_decision 做标签映射, cache_gate_audit 随阶梯传递, 另加 1/64 采样的「闸门失效」告警。
- 6. 数据契约与测试配套: policies/mod.rs 给 CacheCandidate 加 matched_prefix_blocks、给 CacheCandidateProposal 加 worker_queue_limit; policies/cache_aware.rs 抽出 cap_matched_prefix_blocks, 让 token 估计与直方图共用同一 clamp; scoring/mod.rs、bucket_domains.rs、shared_prefill_admission.rs 同步签名与字段。

两条「第二层」的触发条件不同, 这正是本特性成立的原因:

路径	候选池	第二层触发条件
range fallback	整个合法 fleet	池为空——确实没有更好的落点
cache candidates	仅前缀 owner	prefill fleet 内没有任何未排队 worker

若按「候选集为空」触发第二层, 单前缀 owner 的场景会让闸门变成空操作: 闸门拒绝 owner → 集合清空 → 无闸门的一层把同一个排队 owner 放回来, 亲和流量照旧涌向闸门本想避开的地方。

关键源码片段

experimental/sgl-router/src/policies/selection.rs

把闸门审计翻译成指标标签的地方: 新增纯函数 `cache_aware_fallback_decision` (review 修复的核心, `fleet_all_queued` 必须早于容量判断)、`cache_gate_audit` 随选择阶梯传递, 以及「只对真正选出 worker 的选择记账」的门控; 还有 1/64 采样的闸门失效告警。

```

/// 把一次「没有选出 winner」的 Cache-Aware 选择映射到 decision 标签。
/// 纯函数，因此每个边界都能被单元测试钉住，而不是靠调用它的阶梯反推：
///
/// - `queue_gate_rejected == 0`：闸门什么都没摘，候选集变空与闸门无关。
/// - `fleet_all_queued` 在容量问题之前询问：`all_queued` 的语义是
///   「fleet 饱和」，而不是「请求落在哪里」。若先问容量，最坏情况下的
///   饱和信号会被丢掉——一个排队中的 fleet 如果它的 owner 又恰好 KV 不足，
///   就只记 `cache_miss`，满载 fleet 读起来像健康 fleet，
///   正是这个标签要消除的盲区。
/// - `admission_evaluated == 0` 才是「空集由闸门而非容量造成」的判据；
/// 没有它，只要有一个 owner 在排队，容量耗尽就会被记成闸门分流。
fn cache_aware_fallback_decision(
  queue_gate_rejected: u64,
  admission_evaluated: u64,
  fleet_all_queued: bool,
) -> CacheAwareDecision {
  if queue_gate_rejected == 0 {
    return CacheAwareDecision::CacheMiss;
  }
  if fleet_all_queued {
    return CacheAwareDecision::AllQueued;
  }
  if admission_evaluated > 0 {
    // 是容量问题而非闸门：owner 过了闸门，却没过容量准入
    return CacheAwareDecision::CacheMiss;
  }
  CacheAwareDecision::CacheWorkerQueued
}

// 节选自 `Selector::run` 的记账块：只有解析出 worker 的选择才记账。
// 走完所有阶梯仍无 worker 是 503，已由
// `sgl_router_policy_selection_failures_total` 统计；在这里再记一次会破坏
// 「标签之和 = cache-aware 请求速率 - 503」的约定，更糟的是会让一个什么都没
// 选到的请求记成 `cache_worker_queued` 并喂给
// `sgl_router_diverted_overlap_blocks`——没有落地的分流不是闸门代价的证据。
// 这一门控正是 review 意见 1 的修复。
if inputs.policy_kind == PolicyKind::CacheAware && !cache_winner_hit && selected.is_some() {
  let (rejected, evaluated, fleet_all_queued, blocks) =
    self.cache_gate_audit.unwrap_or((0, 0, false, 0));
  let decision = cache_aware_fallback_decision(rejected, evaluated, fleet_all_queued);
  if matches!(decision, CacheAwareDecision::CacheWorkerQueued) {
    // 真正发生分流：存在未排队落点，闸门放弃了 `blocks` 个已匹配前缀块
    inputs
      .metrics
      .observe_diverted_overlap_blocks(&inputs.model_id.0, u64::from(blocks));
  }
  inputs
    .metrics
    .record_cache_aware_decision(&inputs.model_id.0, decision);
}

```

```

}

/// 闸门失效告警按 1/64 采样：它跑在每请求路径上，而触发条件
/// （配了闸门 + 全 fleet 零新鲜负载样本）是稳态的，采样足够暴露问题
/// 又不会刷屏日志。
const QUEUE_GATE_BLIND_LOG_SAMPLE: u64 = 64;
static QUEUE_GATE_BLIND_LOG_COUNTER: AtomicU64 = AtomicU64::new(0);

/// 节选自 `Selector::cache_winner`：闸门逐 worker fail open，当全 fleet
/// 都没有新鲜样本时闸门整体失效，而 `cache_worker_queued` 恒为 0 与
/// 「健康 fleet」无法区分，因此必须显式告警。
if inputs.worker_queue_limit.is_some()
    && !inputs.workers.is_empty()
    && inputs
        .workers
        .iter()
        .all(|worker| snapshot.fresh_load_for_url(&worker.url).is_none())
    && QUEUE_GATE_BLIND_LOG_COUNTER
        .fetch_add(1, AtomicOrdering::Relaxed)
        .is_multiple_of(QUEUE_GATE_BLIND_LOG_SAMPLE)
{
    tracing::warn!(
        model = %inputs.model_id,
        worker_queue_limit = inputs.worker_queue_limit,
        workers = inputs.workers.len(),
        "--worker-queue-limit is set but no worker has a fresh engine load \
sample, so the queue gate is inert. Check that engines advertise a \
load port and publish LoadStat",
    );
}

```

experimental/sgl-router/src/server/metrics.rs

新增 `CacheAwareDecision` 四态标签、`sgl_router_cache_aware_decisions_total` 计数器与 `sgl_router_diverted_overlap_blocks` 直方图（`OVERLAP_BLOCK_BUCKETS`），并在模块文档里定义了各标签的读法；review 的第一条意见正是针对这里的文档不变量。

```

/// `sgl_router_diverted_overlap_blocks` 的分桶上界（单位：块）。
/// 2 的幂直到 8192 块；块大小由引擎配置（常见 16–64 token），
/// 因此这套阶梯覆盖约 16 token 到约 512K token 的「被放弃前缀」。
const OVERLAP_BLOCK_BUCKETS: &[f64] = &[
    1.0, 2.0, 4.0, 8.0, 16.0, 32.0, 64.0, 128.0, 256.0, 512.0, 1024.0, 2048.0, 4096.0, 8192.0,
];

/// 每个 cache-aware prefill 选择的最终决策，只记一次。
/// 标签语义见模块文档。
#[derive(Debug, Clone, Copy)]
pub enum CacheAwareDecision {
    CacheHit,
    CacheMiss,

```

```

    CacheWorkerQueued,
    AllQueued,
}

impl CacheAwareDecision {
    fn as_str(self) -> &'static str {
        match self {
            Self::CacheHit => "cache_hit",
            Self::CacheMiss => "cache_miss",
            Self::CacheWorkerQueued => "cache_worker_queued",
            // 故意不写成 `cache_hit_*`: `decision=~"cache_hit.*"` 的命中率
            // 查询不能把它吸收进去, 否则满载 fleet 会读成健康 fleet。
            Self::AllQueued => "all_queued",
        }
    }
}

// 节选自 `MetricsRegistry`: 决策计数器按 `model_id` + `decision` 建键,
// 一次 cache-aware 选择恰好调用一次, 因此各标签之和就是 cache-aware 的
// 请求速率 (减去记为 503 的选择)。
pub fn record_cache_aware_decision(&self, model_id: &str, decision: CacheAwareDecision) {
    let key = CacheAwareDecisionKey {
        model_id: model_id.to_owned(),
        decision: decision.as_str(),
    };
    let mut guard = self.cache_aware_decisions_total.lock();
    let counter = guard
        .entry(key)
        .or_insert_with(|| Arc::new(AtomicU64::new(0)))
        .clone();
    drop(guard);
    counter.fetch_add(1, Ordering::Relaxed);
}

/// 观测一次闸门分流放弃的匹配前缀深度 (块), 喂给
/// `sgl_router_diverted_overlap_blocks`。只在闸门清空候选集
/// (`cache_worker_queued`) 时记录, 因此直方图度量的是「牺牲」而不是流量;
/// 只摘掉最深 owner、更浅 owner 仍胜出的部分分流不计入这里,
/// 那种情况会被记成 `cache_hit`。
pub fn observe_diverted_overlap_blocks(&self, model_id: &str, blocks: u64) {
    let mut guard = self.diverted_overlap_blocks.lock();
    let hist = guard
        .entry(model_id.to_owned())
        .or_insert_with(|| Histogram::new(OVERLAP_BLOCK_BUCKETS));
    hist.observe(blocks as f64);
}

```

评论区精华

两条 review 意见都来自 ShangmingCai，都在第二个 commit 中修复。

「This invariant doesn't match the code: the recording block in `Selector::run` also runs when the ladder returns `Err`, so a selection that ends in a 503 still books a decision... Should we fix this doc, or gate the record on `selected.is_some()`?」 (`server/metrics.rs:48`)

作者选择门控代码而非改文档：记账块加 `selected.is_some()`，让「没有落地的分流不是闸门代价的证据」这一语义在代码里成立。

「Seems unreachable: `queue_gate_fell_back` re-admits every candidate, so `admission_evaluated > 0` whenever `fleet_all_queued && rejected > 0`... Also means a saturated fleet whose re-admitted owners then fail capacity books `cache_miss`, dropping the saturation signal — intended?」

修复方式是把 `fleet_all_queued` 提到容量判断之前，并把单元测试从 resolver 不可能产生的 `(2, 0, true)` 换到 `(2, 2, true)`，另加三个端到端接线测试。这条讨论的价值在于它证明「纯映射函数 + resolver 审计」这种分层是可测的——测试确实抓到了一条只有在极端状态下才会暴露的错误标签。

风险与影响

- 闸门可能静默失效：`queue_gate_admits` 对无新鲜样本的 worker 一律放行，只要引擎没有发布负载样本，`--worker-queue-limit` 就是空操作。唯一提示是 1/64 采样的 warn，PR body 也承认目前「There is no exported counter for them yet」，闸门拒绝数要等后续 `sgl_router_queue_gate_rejected_total`。
- 阈值与 `dp_size` 的隐式耦合：样本把各 DP rank 的 waiting 求和，而请求只落到其中一个 rank，limit 未随 `--dp-size` 缩放时会系统性偏严或偏松，且只有文档注释提醒。
- 部分分流不可观测：若闸门只摘掉最深 owner 而更浅 owner 仍胜出，记 `cache_hit` 且不进 `sgl_router_diverted_overlap_blocks`；命中率面板可能看不出请求正在离开前缀。
- 每请求的 fleet 级扫描：`fleet_is_all_queued` 对每个 cache-aware 请求做一次全 fleet `all(...)` 遍历，闸门开启后的成本从候选集规模变成 fleet 规模 + 候选集规模，大 fleet 高 QPS 下需要实测。
- 压力比较覆盖面变化：FreshLoadLookup 从「全部候选」改为围绕 `evaluated` 构建，这是有意为之，但闸门开启后 `prefill_pressure_source` 的覆盖率会随闸门结果变化，属于隐式行为差异。
- 影响范围：只作用于 `--policy cache_aware` 的 `prefill` 选择路径；`decode` 选择与 `CapacityFallbackPowerOfTwo` 最后手段不受闸门约束，因此不会出现「无处可去」的请求风暴。未设置 flag 时行为不变 (PR body: Unset, the flag changes nothing)。

关联脉络

本 PR 是「Router 局部性」PR 栈的第 2 层：第 1 层 #39322 抽出 selection seam 并已合入，本 PR 因此能把闸门写成阶梯中的一级而不是围绕阶梯打补丁 (`routes/chat.rs` 从 +112/-7 降到 +13/-0)；第 3 层 #39169 (`saturation pin`、`--saturation-queue-floor`) 与第 4 层

#39170 (candidate sampling、`--min-load-choices`) 以本分支为 base，需各自 rebase 后再评审。数据来源上依赖 #34608 的基础负载采样 (`num_waiting_reqs`)。配置与指标风格则延续同仓库的 Router 演进线：#39000/#39001 的 fleet-wide sampling contract 建立了「配置面 + 每请求校验 + 指标」的范式，#39015 的关闭排空配置面与 #39014 的在途计数同样在 `config/` 与 `server/metrics.rs` 上扩展可观测性。整体趋势清晰：Router 正在把「靠经验调参」换成「有决策标签、有代价直方图、有失效告警」的可运营路由层。