PR #25062 完整报告

sgl-project/sglang

[PD Disaggregation] Fix priority scheduling in PD disaggregation mode

合并时间：2026-05-14 02:43

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25062

执行摘要

- 一句话：修复 PD 分离模式下优先级调度静默失效问题
- 推荐动作：建议阅读此 PR，尤其关注 _add_request_to_queue 的控制流调整和 pop_preallocated 中排序时机的决策。设计上值得注意的点：将优先级设置与模式分支解耦，以及在索引操作前排序以保证一致性。

功能与动机

根据 PR 描述和关联 Issue #25060，使用 --enable-priority-scheduling 与 prefill-decode disaggregation 且未设置默认优先级时，优先级调度静默失效。

实现拆解

1. scheduler.py— 将 _set_or_validate_priority 调用提前到 _add_request_to_queue 中所有 disaggregation 模式分支之前，确保默认优先级在 PREFILL 和 DECODE 模式下也被分配。
2. disaggregation/decode.py get_new_prebuilt_batch— 在从等待队列选择请求前调用 policy.calc_priority，与 prefill 路径行为一致。
3. disaggregation/decode.py pop_preallocated— 在索引簿记（abort 循环）之前对 self.queue 按优先级排序，避免因排序时机错误导致健康请求被丢弃。
4. 新增单元测试test_priority_scheduling_disaggregation.py — 覆盖 PREFILL 和 DECODE 模式的默认优先级分配、优先级禁止时的 abort 行为以及 decode 预分配队列的优先级排序。

关键文件：

- test/registered/unit/managers/test_priority_scheduling_disaggregation.py（模块 测试；类别 test；类型 test-coverage；符号 TestDisaggregationPriorityQueueing, _new_scheduler, _new_req, test_prefill_mode_assigns_default_priority_before_bootstrap_queue）：新增单元测试，覆盖 PD 分离模式下优先级调度的核心场景，确保修复有效且防止回归。
- python/sglang/srt/disaggregation/decode.py（模块 解码器；类别 source；类型 core-logic；符号 pop_preallocated, get_new_prebuilt_batch）：核心修复文件，修改了 pop_preallocated 中的排序时机和在 get_new_prebuilt_batch 中添加优先级计算。

- python/sglang/srt/managers/scheduler.py（模块 调度器；类别 source；类型 core-logic；符号 _add_request_to_queue, _set_or_validate_priority）：将优先级设置 / 验证提前到所有模式分支之前，确保 PREFILL 和 DECODE 模式也能分配默认优先级。

关键符号：_add_request_to_queue, _set_or_validate_priority, pop_preallocated, get_new_prebuilt_batch

关键源码片段

test/registered/unit/managers/test_priority_scheduling_disaggregation.py

新增单元测试，覆盖 PD 分离模式下优先级调度的核心场景，确保修复有效且防止回归。

```
import sys
import unittest
from types import SimpleNamespace
from unittest.mock import MagicMock, patch

import torch

from sglang.srt.disaggregation.decode import ( # noqa: E402
    DecodePreallocQueue,
    SchedulerDisaggregationDecodeMixin,
)
from sglang.srt.disaggregation.utils import DisaggregationMode # noqa: E402
from sglang.srt.managers.schedule_batch import FINISH_ABORT # noqa: E402
from sglang.srt.managers.scheduler import Scheduler # noqa: E402
from sglang.test.ci.ci_register import register_cuda_ci

register_cuda_ci(est_time=5, suite="stage-a-test-1-gpu-small")


class TestDisaggregationPriorityQueueing(unittest.TestCase):
    def _new_scheduler(self, disaggregation_mode: DisaggregationMode) -> Scheduler:
        scheduler = Scheduler.__new__(Scheduler)
        scheduler.disaggregation_mode = disaggregation_mode
        scheduler.enable_priority_scheduling = True
        scheduler.schedule_low_priority_values_first = False
        scheduler.abort_on_priority_when_disabled = False
        scheduler.waiting_queue = []
        scheduler._prefetch_kvcache = MagicMock()
        scheduler._abort_on_queued_limit = MagicMock(return_value=False)
        scheduler.model_config = SimpleNamespace(num_key_value_heads=8)
        scheduler.disagg_prefill_bootstrap_queue = MagicMock()
        scheduler.disagg_decode_prealloc_queue = MagicMock()
        scheduler.send_to_tokenizer = MagicMock()
        return scheduler

    def _new_req(self, priority=None):
        req = MagicMock()
```

```
        req.priority = priority
        req.rid = "req"
        req.time_stats = MagicMock()
        req.time_stats.trace_ctx = MagicMock()
        return req

    def test_prefill_mode_assigns_default_priority_before_bootstrap_queue(self):
        # 测试 PREFILL 模式下，priority 为 None 的请求被分配默认优先级后再加入队列
        scheduler = self._new_scheduler(DisaggregationMode.PREFILL)
        req = self._new_req(priority=None)

        scheduler._add_request_to_queue(req)

        self.assertEqual(req.priority, -sys.maxsize - 1)
        scheduler.disagg_prefill_bootstrap_queue.add.assert_called_once_with(req, 8)
        req.time_stats.set_prefill_bootstrap_queue_entry_time.assert_called_once()

    def test_decode_mode_assigns_default_priority_before_prealloc_queue(self):
        # 测试 DECODE 模式下，priority 为 None 的请求被分配默认优先级后再加入队列
        scheduler = self._new_scheduler(DisaggregationMode.DECODE)
        req = self._new_req(priority=None)

        scheduler._add_request_to_queue(req)

        self.assertEqual(req.priority, -sys.maxsize - 1)
        scheduler.disagg_decode_prealloc_queue.add.assert_called_once_with(
            req, is_retracted=False
        )
        req.time_stats.set_decode_prealloc_queue_entry_time.assert_called_once()

    def test_priority_disabled_abort_validation_applies_to_decode_mode(self):
        # 测试优先级调度禁用且设置了 abort_on_priority_when_disabled 时，请求被拒绝
        scheduler = self._new_scheduler(DisaggregationMode.DECODE)
        scheduler.enable_priority_scheduling = False
        scheduler.abort_on_priority_when_disabled = True
        req = self._new_req(priority=10)

        scheduler._add_request_to_queue(req)

        scheduler.disagg_decode_prealloc_queue.add.assert_not_called()
        scheduler.send_to_tokenizer.send_output.assert_called_once()
        req.time_stats.trace_ctx.abort.assert_called_once()
```

python/sglang/srt/disaggregation/decode.py

核心修复文件，修改了 pop_preallocated 中的排序时机和在 get_new_prebuilt_batch 中添加优先级计算。

```
# pop_preallocated 中：在 abort 循环和预分配循环之前按优先级排序队列
def pop_preallocated(self, rids_to_check=None):
```

```
    # ... 前面计算 allocatable_tokens 的代码 ...

    # 在索引操作之前按优先级排序，保证 abort 扫描和预分配使用相同顺序
    if self.scheduler.enable_priority_scheduling:
        priority_sign = 1 if self.scheduler.schedule_low_priority_values_first else -1
        self.queue.sort(key=lambda r: r.req.priority * priority_sign)

    # 首先移除失败的请求
    for i, decode_req in enumerate(self.queue):
        if rids_to_check is not None and decode_req.req.rid not in rids_to_check:
            continue
        if isinstance(decode_req.req.finished_reason, FINISH_ABORT):
            self.scheduler.stream_output([decode_req.req], decode_req.req.return_logprob)
            failed_reqs.append(decode_req)
            indices_to_remove.add(i)
    # ... 后续预分配逻辑 ...

# get_new_prebuilt_batch 中：在从等待队列构建 batch 之前计算优先级
def get_new_prebuilt_batch(self):
    if self.grammar_manager.has_waiting_grammars():
        ready_grammar_requests = self.grammar_manager.get_ready_grammar_requests()
        for req in ready_grammar_requests:
            self._add_request_to_queue(req)

    if len(self.waiting_queue) == 0:
        return None

    # 添加：在构建 batch 前计算优先级，与 prefill 路径一致
    if self.enable_priority_scheduling:
        self.policy.calc_priority(self.waiting_queue, self.running_batch)

    curr_batch_size = self.running_batch.batch_size()
    batch_size = min(self.req_to_token_pool.size, self.max_running_requests)
    # ... 继续构建 batch ...
```

评论区精华

机器人审查建议在优先级排序中添加次级排序键（如进入队列时间）以确保相同优先级的确定性顺序，但作者未采纳该建议。两位人工审核者（ishandhanani, ShangmingCai）均批准了此 PR，ShangmingCai 表示“Looks clean”。

- 建议在优先级排序中添加次级排序键以确保确定性 (design): 未被采纳，PR 中未修改。

风险与影响

- 风险：核心调度路径修改可能影响其他模式，但改动集中且已有测试覆盖。在 decode 路径添加排序可能引入轻微性能开销，但仅在启用优先级调度时生效，且队列规模通常较小。未引入不兼容变更。

- 影响：影响范围：使用 PD 分离模式且启用优先级调度的用户。影响程度：功能修复，确保优先级调度正确工作，无破坏性变更。新增的测试为后续维护提供了安全网。
- 风险标记：核心路径变更，优先级排序性能影响

关联脉络

- PR #22536 [Disagg][NIXL] Add staging buffer support for heterogeneous TP KV transfer: 同属 PD 分离功能线，涉及 disaggregation 模式的通信优化。