PR #29408 完整报告

sgl-project/sglang

Avoid implicit field-based side channel in Scheduler planning

合并时间：2026-07-10 08:55

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/29408

执行摘要

- 一句话：消除调度器 batch 决策中的隐式副作用通道
- 推荐动作：值得精读。展示了系统化消除模块内部隐式耦合的设计模式，以及如何用 guard 测试强制执行架构恒量。对于从事调度系统或中间件开发的工程师有借鉴意义。

功能与动机

PR body 指出：get_next_batch_to_run 及其调用树（包括 disaggregation 的对应方法）使用了 self.cur_batch / self.last_batch / self.running_batch 作为隐式的输入 / 输出副作用通道。本 PR 将这些读写显式化：batch 状态通过参数传入，并通过 NextBatchPlan 结构体返回。

实现拆解

1. 引入 NextBatchPlan 结构体：在 schedule_batch.py 中新增 NextBatchPlan 作为 msgspec.Struct，包含 batch_to_run 和 running_batch 字段，用于显式封装调度决策的输出。
2. 改造核心决策方法：修改 get_next_batch_to_run、get_new_batch_prefill、_abort_on_running_timeout、is_disable_overlap_for_batch 等方法的签名，使其接受 running_batch 和 / 或 last_batch 作为参数，并返回 NextBatchPlan 而不是直接修改 self。方法内部不再读取或写入 self.running_batch / self.last_batch。
3. 改造 disaggregation 路径：对 get_next_disagg_prefill_batch_to_run、get_next_disagg_decode_batch_to_run、process_prefill_chunk、get_new_prebuilt_batch 等做类似改造，将隐式状态改为参数传递。
4. 更新所有调用点：在 event_loop_normal、event_loop_overlap、event_loop_pp、MLX 混合等事件循环中，将原来的无参调用改为传入 self.running_batch 和 self.last_batch，并将返回的 plan.running_batch 写回 self.running_batch，确保行为等价。
5. 新增 guard 测试：在 test_scheduler_decision_batch_params.py 中，通过 inspect.getsource 检查所有决策方法的源码，断言它们不包含 self.running_batch、self.last_batch、self.cur_batch 等 token，将无隐藏通道的约束转化为自动化测试。
6. 已知残余：DLLM 路径（get_new_batch_dllm）仍包含 self.running_batch 的写入，作为已知残余留待后续处理，排除在 guard 测试范围之外。

关键文件：

- python/sglang/srt/managers/scheduler.py（模块 调度器；类别 source；类型 core-logic；符号 _abort_on_running_timeout, is_disable_overlap_for_batch, get_next_batch_to_run, get_new_batch_prefill）：核心调度器，所有 batch 决策方法在此重构
- python/sglang/srt/managers/schedule_batch.py（模块 调度批处理；类别 source；类型 core-logic；符号 NextBatchPlan）：定义了 NextBatchPlan 结构体，是显式状态传递的载体
- test/registered/unit/managers/test_scheduler_decision_batch_params.py（模块 测试；类别 test；类型 test-coverage；符号 TestDecisionMethodsHaveNoHiddenBatchChannel, test_decision_methods_take_batches_as_params_not_self）：guard 测试，确保决策方法不再引用 self.*_batch
- python/sglang/srt/disaggregation/decode.py（模块 分离解码；类别 source；类型 core-logic；符号 get_new_prebuilt_batch, get_next_disagg_decode_batch_to_run）：分离解码路径的调度决策方法重构
- python/sglang/srt/disaggregation/prefill.py（模块 分离预填；类别 source；类型 core-logic；符号 process_prefill_chunk, get_next_disagg_prefill_batch_to_run）：分离预填路径的调度决策方法重构

关键符号：NextBatchPlan, get_next_batch_to_run, get_new_batch_prefill, _abort_on_running_timeout, is_disable_overlap_for_batch, get_next_disagg_prefill_batch_to_run, process_prefill_chunk, get_next_disagg_decode_batch_to_run, get_new_prebuilt_batch

关键源码片段

python/sglang/srt/managers/scheduler.py

核心调度器，所有 batch 决策方法在此重构

```
# _abort_on_running_timeout 现在接受 running_batch 参数，不再访问 self.running_batch
def _abort_on_running_timeout(self, running_batch: ScheduleBatch):
    # NOTE: should be called before a batch is launched.
    timeout_s = envs.SGLANG_REQ_RUNNING_TIMEOUT.get()
    if timeout_s <= 0:
        return
    if running_batch.is_empty():
        return

    deadline = time.perf_counter() - timeout_s
    for req in running_batch.reqs:
        if not req.finished() and 0 < req.time_stats.forward_entry_time < deadline:
            req.to_finish = FINISH_ABORT(
                "Request running timeout reached.", HTTPStatus.SERVICE_UNAVAILABLE
            )

# event_loop 中的调用变化示例
# 原：batch = self.get_next_batch_to_run()
```

```
# 新：
plan = self.get_next_batch_to_run(
    running_batch=self.running_batch, last_batch=self.last_batch
)
self.running_batch = plan.running_batch
batch = plan.batch_to_run
```

test/registered/unit/managers/test_scheduler_decision_batch_params.py

guard 测试，确保决策方法不再引用 self.*_batch

```
# guard 测试：检查决策方法是否引用了 self.*_batch
class TestDecisionMethodsHaveNoHiddenBatchChannel(unittest.TestCase):
    def test_decision_methods_take_batches_as_params_not_self(self):
        # The batch decision tree must receive running/last batch as params, never via self.*
        for method in DECISION_METHODS:
            source = inspect.getsource(inspect.unwrap(method))
            self.assertIn(
                f"def {method.__name__}", source,
                msg=f"failed to read the real source of {method.__qualname__}",
            )
            for token in FORBIDDEN_TOKENS:
                self.assertNotIn(
                    token, source,
                    msg=(
                        f"{method.__qualname__} references {token}; pass the batch "
                        "explicitly and return it via NextBatchPlan instead."
                    ),
                )
```

评论区精华

review 中 gemini-code-assist[bot] 指出 get_new_batch_dllm 仍然赋值 self.running_batch，建议重构为返回 NextBatchPlan 并加入 guard 测试。PR 作者已在 PR body 中将其列为已知残余（Known residual），说明 DLLM prefill helpers 仍然内部读取 self.running_batch，但行为未变，完全线程化留作后续跟进。该讨论的结论是“已确认，后续处理”。

- DLLM 路径未完全重构 (design): PR 作者在 PR body 中将其列为已知残余，说明 DLLM 路径暂时保留原样，行为未变，完全线程化留作后续跟进。

风险与影响

- 风险：核心调度路径的修改可能引入回归，但经过 Codex 等价性审计和新增的 guard 测试，风险可控。主要风险在于：DLLM 路径未被覆盖，可能在未来不小心引入不一致；对 pipeline-parallel 和 disaggregation 路径的改动需要仔细验证，但这些路径有相应的测试；没有性能影响，因为只是将函数参数传递从隐式改为显式。
- 影响：对用户透明，无功能或性能变化。对开发者而言，这一重构使得调度决策方法更加纯净，为后续拆分为无状态的 SchedulerPlanner、支持更复杂的调度策略铺平了道路。团队内需注意后续开发不应重新引入隐式状态。

- 风险标记：核心路径变更，未完全消除侧通道 (DLLM)，引入回归风险

关联脉络

- 暂无明显关联 PR