

MoonWheelKit 项目申报书

分层时间轮与确定性虚拟时钟调度基础库

一、基本信息

项目名称	MoonWheelKit: 面向 MoonBit 的分层时间轮与确定性虚拟时钟调度基础库
参赛者	孙加豪
联系方式	【提交前填写: 手机、邮箱】
GitHub 仓库	https://github.com/brother-666/MoonWheelKit
GitLink 仓库	https://www.gitlink.org.cn/brother666/MoonWheelKit
项目方向	MoonBit 基础生态库 / 定时调度 / 确定性测试基础设施
是否为移植项目	否
当前基础	15+ 次有效提交; 21 项测试; 四后端验证; GitHub CI 已通过

二、项目简介

MoonWheelKit 面向需要管理大量截止时间的 MoonBit 程序, 提供可嵌入的分层时间轮和虚拟时钟调度内核。项目不读取系统时钟, 不创建线程, 也不绑定浏览器、Node.js 或操作系统接口; 调用方通过推进逻辑 tick 驱动调度, 因此同一操作序列在 Native、JavaScript、WebAssembly 和 Wasm-GC 后端具有一致的触发结果。

项目适用于网关超时、缓存失效、重试队列、游戏逻辑、离散事件模拟、测试替身和异步运行时适配。它解决的不是单个任务如何 sleep, 而是大量未来截止时间如何分层组织、低成本取消、迟到后如何追赶以及调度状态如何恢复。

三、拟解决的问题与生态价值

MoonBit 已有异步运行时和 sleep、timeout、retry 等能力, 但这些接口面向真实事件循环, 难以直接承担可重复的虚拟时间测试、批量截止时间管理和调度状态恢复。业务若自行维护排序数组或零散 timer, 取消通常需要扫描, 迟到与周期漂移语义也容易因后端而异。

MoonWheelKit 把时间轮作为独立基础设施交付: 上层可接入真实时钟、游戏帧或模拟时钟, 核心始终保持无 IO、后端中立。库同时公开统计、校验、快照和 JSON 证据, 便于测试框架、调试器和教学可视化复用。

四、核心功能范围

1. 可配置层数、每层槽数、tick 单位和最大追赶次数的分层时间轮。
2. 单次、固定延迟、固定频率三类任务; 相同截止时间保持稳定顺序。
3. 基于 timer id 与 generation 的 O(1) 表查找取消、重排和陈旧引用失效。
4. 逐 tick 精确推进, 以及不扫描跳过区间的迟到推进和 lateness 报告。
5. 固定频率任务有界追赶, 长时间暂停后不会产生无上限任务风暴。
6. 逻辑快照、确定性恢复、运行统计、JSON 输出、不变量校验和桶清理。

五、社区查重与边界

2026 年 6 月 30 日以 timing wheel、hierarchical timer、virtual time scheduler、cron、delayed task 等关键词检索 Mooncakes。moonbitlang/async 提供真实事件循环、sleep、timeout 和 retry; peter-jerry-ye/async 提供 promise、event loop、channel 与 IO; f4ah6o/mhx 的 delay、throttle、debounce 面向前端交互; dowliness/moondsp 的调度面向音频领域。

本项目不替代异步运行时、cron 解析器或持久化数据库。检索未发现以“分层槽位 + generation 失效 + 显式虚拟时间 + 有界迟到追赶 + 快照回放”为组合定位的通用 MoonBit 包。相关链接和差异逐项记录在 docs/RELATED_WORK.md。

六、技术设计

设每层槽数为 S ，第 L 层单槽跨度为 S^L tick。新任务进入能够覆盖其 deadline 差值的最低层；时间推进到层级边界时，高层槽中的有效引用逐级下放，临近任务最终进入第 0 层触发。

任务真值只保存在中心 Map 中，槽位仅保存 timer id 与 generation。取消或重排只修改任务状态和 generation，无需在线性桶中查找旧节点；扫描时 generation 不匹配的引用会自然失效，compact 可重建所有有效桶。

advance_to 用于精确逐 tick 观察；advance_late_to 直接跳到目标时间。固定延迟按“实际观察时间 + period”续排，固定频率按“原计划时间 + period”续排，并受 max_catch_up 限制。快照记录几何配置、时钟、计数器与任务状态，恢复后继续产生相同事件序列。

七、创新点

1. 将高吞吐时间轮与可测试虚拟时间统一为一个后端中立内核，而不是事件循环内部不可复用的实现。
2. generation 失效机制把取消、重排从槽位删除中解耦，并通过显式压缩和不变量校验给出可审计状态。
3. 同时定义精确推进和迟到推进，明确区分 FixedDelay 与 FixedRate 漂移，并用有界追赶处理恢复任务风暴。
4. 快照、恢复、稳定排序和四后端差异测试共同保证可重复性，可用于模拟器、测试框架和运行时适配。

八、当前成果与验证

仓库按功能形成 15 次有效提交，覆盖工程骨架、模型、时间轮、基础测试、级联与周期测试、诊断、快照、迟到语义、CLI、查重文档、万级工作负载、README、CI 和公开协作跟踪。

21 项确定性测试覆盖精确截止时间、零延迟、稳定顺序、取消、重排、跨层级联、周期任务、迟到、有界追赶、压缩、统计、校验和快照恢复。默认、JS、Wasm、Wasm-GC 四组测试均通过，GitHub Actions 首次运行成功 (run 28453414903)。

bench/main 确定性调度 10,000 个任务并推进 4,096 tick，结果为 scheduled=10000、fired=10000、pending=0、validation_issues=0。仓库不虚构跨机器性能比例，后续按固定环境发布带日期的测量记录。

九、已知限制

当前时间值使用 Int tick，payload 为 String；快照已可导出 JSON，但尚未提供版本化 JSON 解析器。核心不主动读取真实时钟，也不提供线程安全承诺；这些能力应由适配层按目标后端实现。

十、后续计划

1. 0.2: 时钟适配 trait、批量调度与取消、版本化快照解析、模型对照随机测试。
2. 0.3: 排序数组与二叉堆基线，四后端定期性能报告，聚集和长尾截止时间工作负载。
3. 0.4: 合并窗口、任务组取消、逐任务追赶策略和可注入的确定性抖动策略。

十一、交付物与验收标准

源码交付包括公开 API、21 项测试、CLI 场景、万级确定性工作负载、四后端 CI、架构与语义文档、生态差异说明、变更日志和公开议题清单。

阶段验收以可运行证据为准：moon fmt --check 与 moon check 无告警；四后端测试结果一致；快照恢复前后事件序列一致；万级工作负载无漏触发且校验结果为零。

开发过程在公开仓库持续进行，以功能提交、Issue、Pull Request、CHANGELOG 和带日期的性能记录保持可追踪，不以一次性上传代替开源协作。