Non-Atomic Blockchain Transaction Identification Website
项目设计报告

2025 年 12 月 20 日

目录

1 引言

随着去中心化金融（DeFi）的快速发展，区块链上的交易活动日益频繁。在这些交易中，非原子性套利是一种常见的现象，即交易者利用不同交易平台之间的价格差异进行套利操作。这类操作通常涉及多个独立但相关的交易，而非在一个原子操作中完成。

本项目的目标是建立一个网站平台，专门用于识别和分析此类非原子性套利交易。通过收集来自Uniswap和Binance等交易平台的历史数据，系统可以检测跨平台的价格差异，并发现潜在的套利机会。这对于研究人员、交易员和监管机构来说具有重要意义。

2 系统架构

2.1 总体架构

系统采用前后端分离的现代化架构（如图所示），确保了各模块的独立性和可扩展性。这种设计使得开发、测试和部署更加灵活高效。

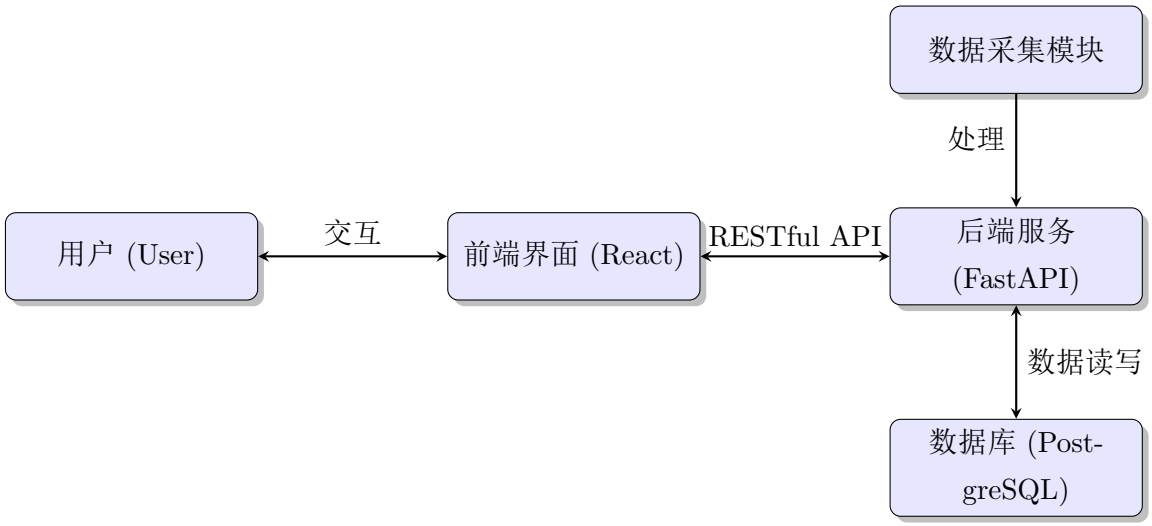

图 1: 系统总体架构图 (优化布局)

系统的核心组件包括：

1. **数据采集模块**：负责从各种交易平台（如 Uniswap、Binance）收集历史交易数据。
2. **后端服务**：基于 FastAPI 框架，提供高性能的 RESTful API 接口，负责业务逻辑处理和数据分析。
3. **数据库层**：使用 PostgreSQL 关系型数据库，高效存储原始交易数据和分析计算后的结果。
4. **前端界面**：基于 React 框架构建的单页应用（SPA），提供直观的数据展示和用户交互功能。
5. **可视化组件**：集成在前端，用于展示价格趋势、套利机会等关键信息图表。

整个系统通过 Docker 容器化部署，实现了环境隔离，便于维护、迁移和水平扩展。

2.2 技术栈选择

为了确保系统的高性能、可靠性和可维护性，我们选择了业界主流且成熟的技术栈。

2.2.1 后端技术

- **FastAPI**：一个现代、高性能的 Python Web 框架，自带异步支持和自动 API 文档生成功能。
- **SQLAlchemy**：强大的 ORM (对象关系映射) 工具，能够简化数据库操作，提高开发效率。
- **PostgreSQL**：功能强大、稳定可靠的开源关系型数据库，非常适合存储结构化数据。
- **Docker**：领先的容器化平台，用于打包、分发和运行应用，确保了开发与生产环境的一致性。

2.2.2 前端技术

- **React**：由 Facebook 推出的用于构建用户界面的 JavaScript 库，以其组件化和高性能而闻名。
- **CSS3**：层叠样式表语言，用于美化界面外观，实现丰富的视觉效果。
- **HTML5**：现代网页标记语言，用于构建网页的基础结构。
- **Docker**：同样用于前端应用的容器化部署，简化了 Nginx 等服务的配置。

3 核心功能实现

3.1 数据处理与分析

系统的核心在于对海量交易数据的处理和分析能力，特别是精准识别非原子性套利机会。

3.1.1 数据模型设计

为了结构化地存储数据，我们设计了以下核心数据模型：

- `UniswapSwap`：存储 Uniswap V3 的 Swap 事件数据。
- `BinanceTrade`：存储币安（Binance）的交易数据。
- `ArbitrageOpportunity`：存储最终识别出的套利机会的详细信息。
- `ArbitrageOpportunityMinute`：按分钟粒度预计算的套利机会，用于快速生成前端图表。

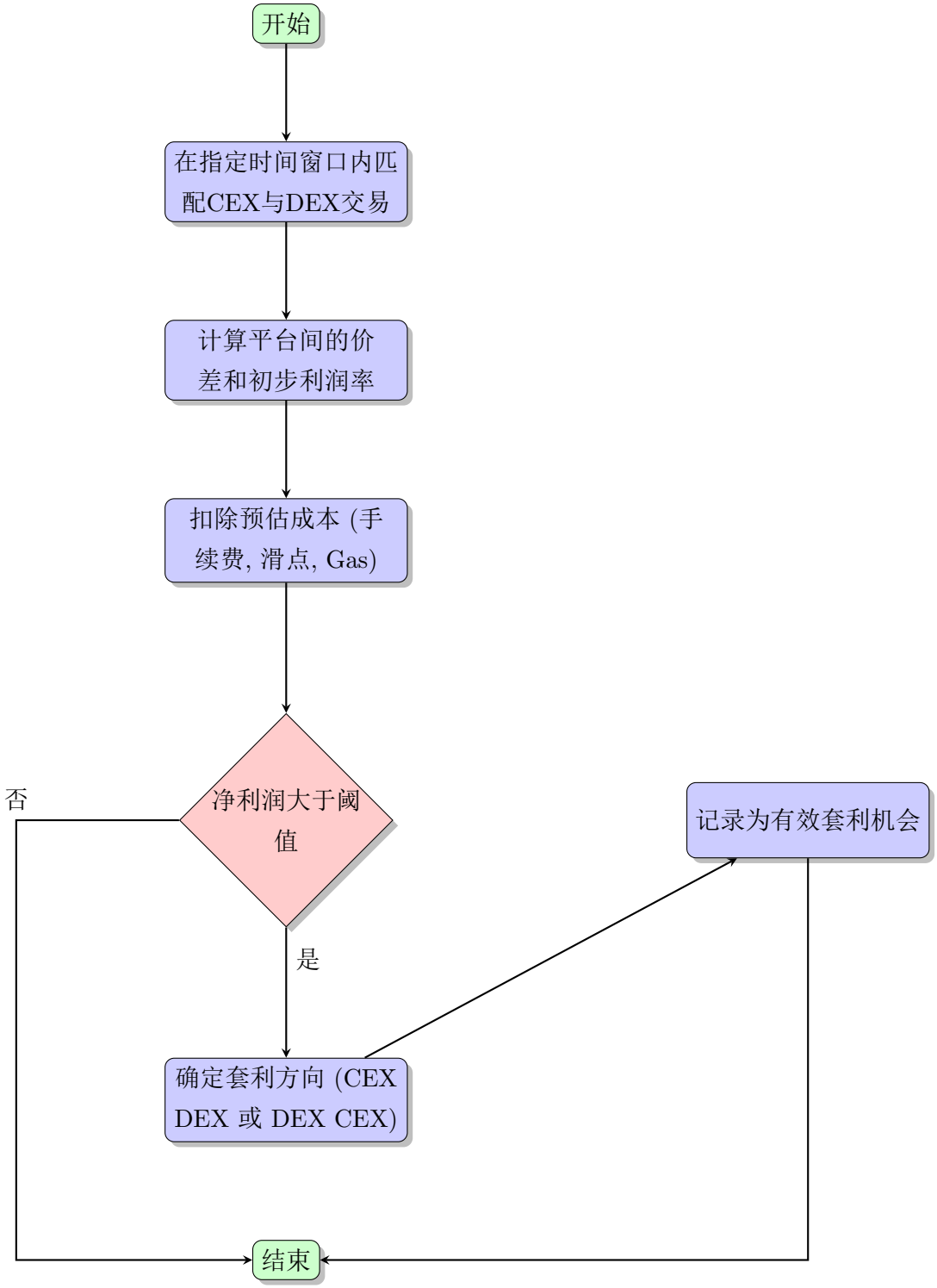

图 2: 套利检测算法流程图

3.1.2 套利检测算法

我们设计并实现了一套复杂的多阶段套利检测算法，其核心流程如图 2 所示。
算法主要考虑以下因素：

- **时间窗口匹配：**在可配置的时间范围（例如数秒内）寻找来自不同平台的、方向相反的交易。

- **价格差异计算**：精确计算两个平台间的价差和潜在利润率。
- **成本考虑**：综合考虑交易手续费、网络Gas费用和可能的滑点，估算净利润。
- **方向判断**：根据价格高低确定最佳的套利路径（例如，从币安买入，转到Uniswap卖出）。

3.2 后端API设计

后端基于 FastAPI 提供了清晰、规范的 RESTful API 端点。

后端核心 API 概览

健康检查接口

- `GET /api/health`：检查后端服务的运行状态。
- `GET /api/db-check`：检查与数据库的连接状态。

价格数据接口

- `GET /api/price-data`：获取 Uniswap 和 Binance 的历史价格数据，按天聚合为 OHLC 格式，用于K线图展示。

套利分析接口

- `GET /api/arbitrage/statistics`：获取套利机会的宏观统计信息（如总次数、总利润）。
- `GET /api/arbitrage/behaviors`：分页获取已识别的详细套利行为列表。
- `GET /api/arbitrage/opportunities`：获取按分钟预计算的套利机会数据，用于前端图表可视化。

3.3 前端界面实现

前端界面设计简洁、直观，主要包含三个核心页面：

3.3.1 信息页面 (`/info`)

作为系统的首页，展示项目的基本介绍、功能说明和使用指南，帮助用户快速了解平台。

3.3.2 价格仪表板 (`/price-dashboard`)

该页面以交互式K线图的形式，并列展示 Uniswap 和 Binance 的历史价格数据。用户可以通过缩放、平移等操作，直观地对比不同市场的价格走势，发现潜在的价格差异。

3.3.3 套利分析 (`/arbitrage-analysis`)

这是系统的核心数据展示页面，包含了丰富的套利分析结果：

- **统计卡片**：醒目地展示总套利机会数量、累计利润、平均利润率等关键指标。
- **套利行为列表**：以表格形式详细列出每一次检测到的套利行为，包括时间、方向、交易对、利润等。
- **套利机会图表**：通过可视化图表（如时间序列图）展示套利机会随时间的分布和强度。

4 部署方案

4.1 容器化部署

为简化部署流程并保证环境一致性，系统整体采用 Docker 容器化方案，通过 `docker-compose` 进行多服务编排：

- **nginx**：作为反向代理服务器，负责请求分发和静态文件服务。
- **frontend**：前端 React 应用的容器。
- **backend**：后端 FastAPI 服务的容器。
- **db**：PostgreSQL 数据库服务的容器，并配置数据卷持久化存储。

4.2 环境配置

通过 `.env` 文件集中管理所有环境变量（如数据库地址、用户名、密码等）。这种方式将配置与代码分离，不仅方便在不同环境（开发、测试、生产）中切换，也极大地提高了敏感信息的安全性。

5 总结与展望

未来，我们计划从以下几个方向对项目进行迭代和优化：

- **扩展数据源**：接入更多主流的中心化（CEX）和去中心化（DEX）交易平台。
- **实现实时处理**：引入消息队列和流处理技术（如 Kafka, Flink），实现对交易数据的实时监控和套利机会的即时发现。
- **引入智能算法**：探索使用机器学习或深度学习模型，根据历史数据预测未来可能出现的套利机会窗口。
- **增强数据可视化**：提供更多维度、更具深度的交互式可视化图表，帮助用户从不同角度分析套利模式。