ICP区块链开发进阶课程

5. 实例项目详解 - Tip Jar

主讲：Paul Liu - DFINITY 工程师

Tip Jar - 向 Canister 捐赠 Cycle

https://tipjar.rocks

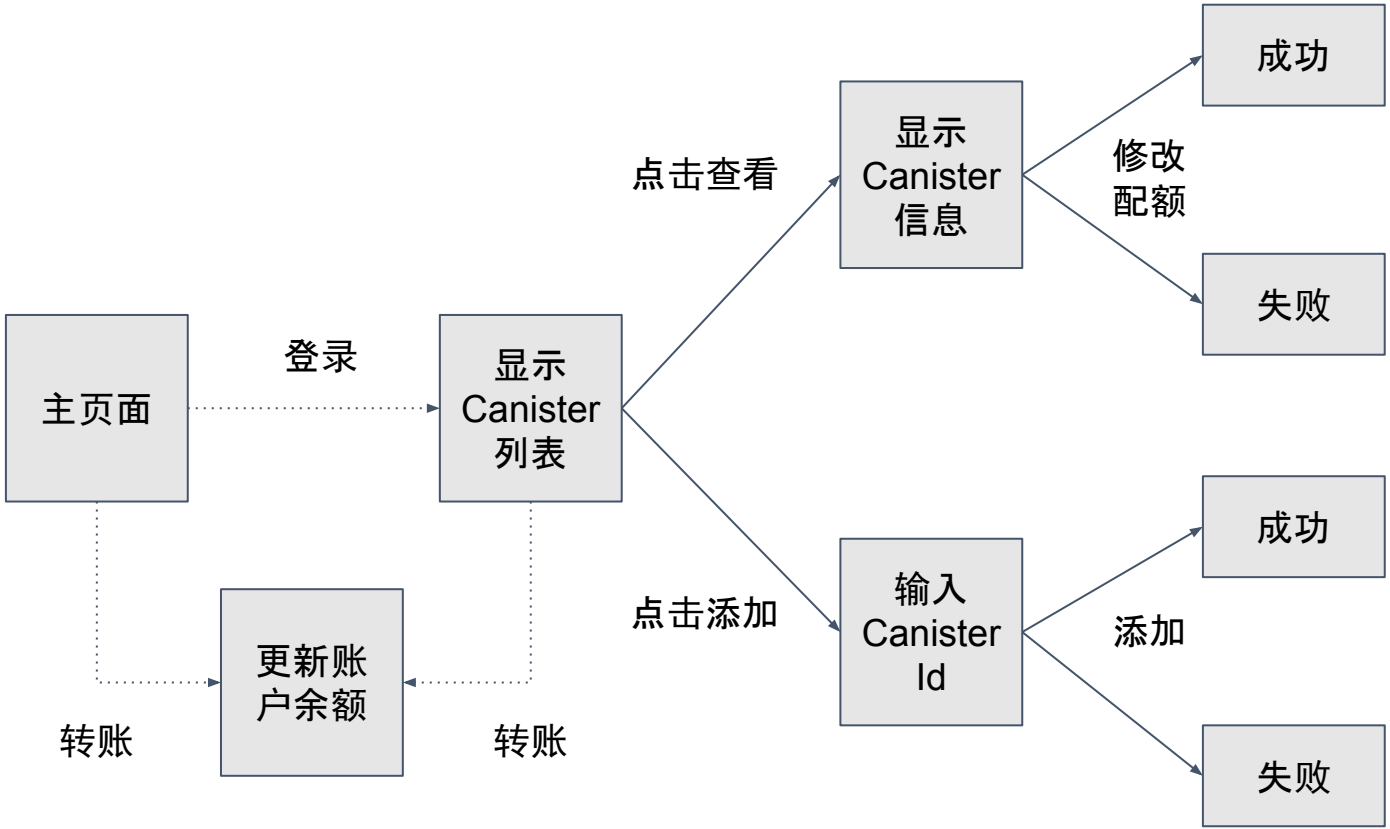
主要功能

1. 用户可以选择向哪些 Canister 捐赠 Cycle
2. Canister 可以接收来自多个用户的捐赠
3. 捐赠不是一次性的，而是根据 Canister 当前 Cycle 余额的变化动态调整
4. 用户可以随时改变对 Canister 捐赠的配额

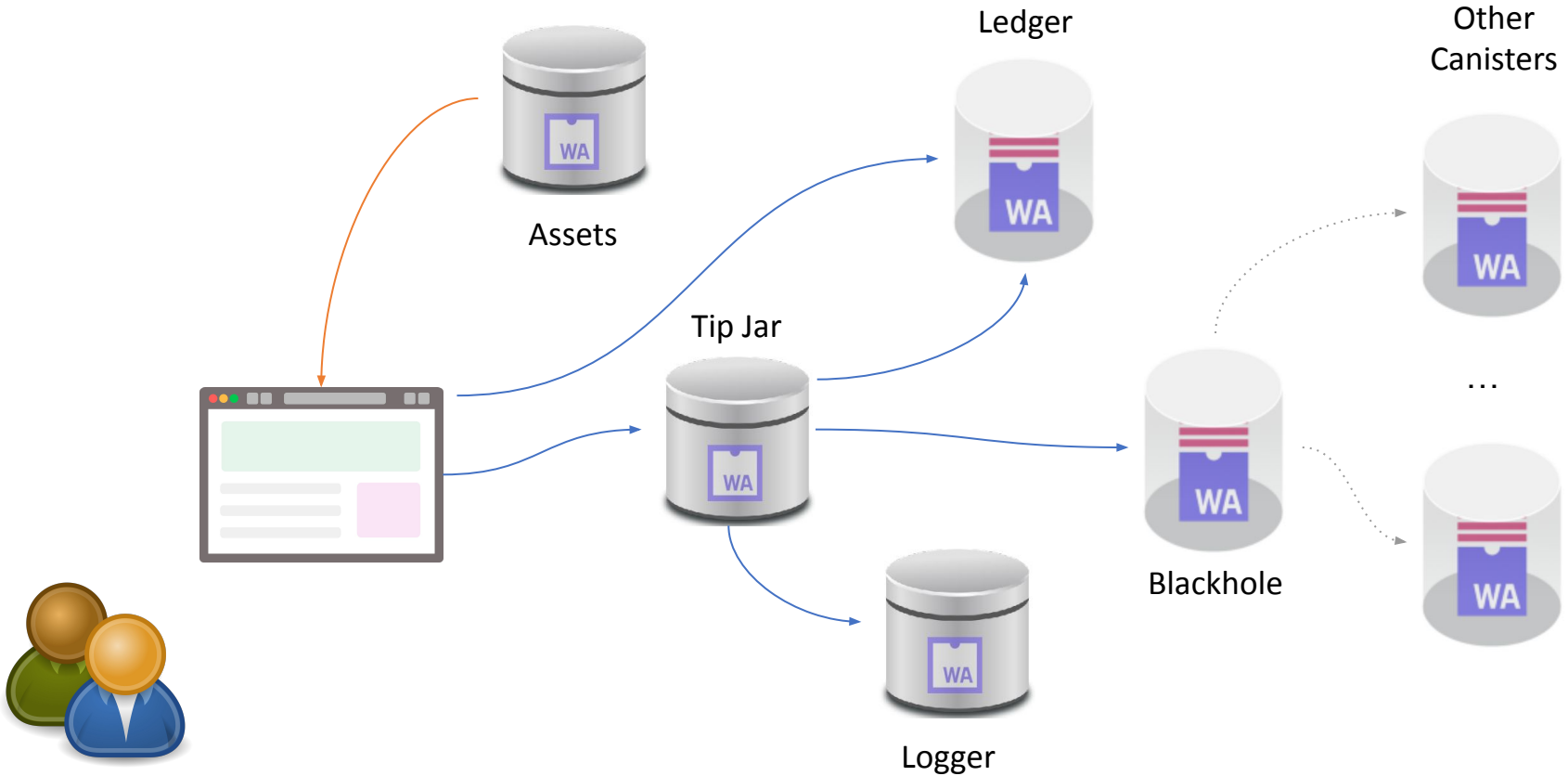
监测 Canister 的 Cycle 余额

- 被监测对象需要把 blackhole e3mmv-5qaaa-aaaah-aadma-cai 加入到控制者名单
- Tip Jar 定期轮询被监测对象的 canister 状态 (由心跳触发)

用户操作流程



项目架构

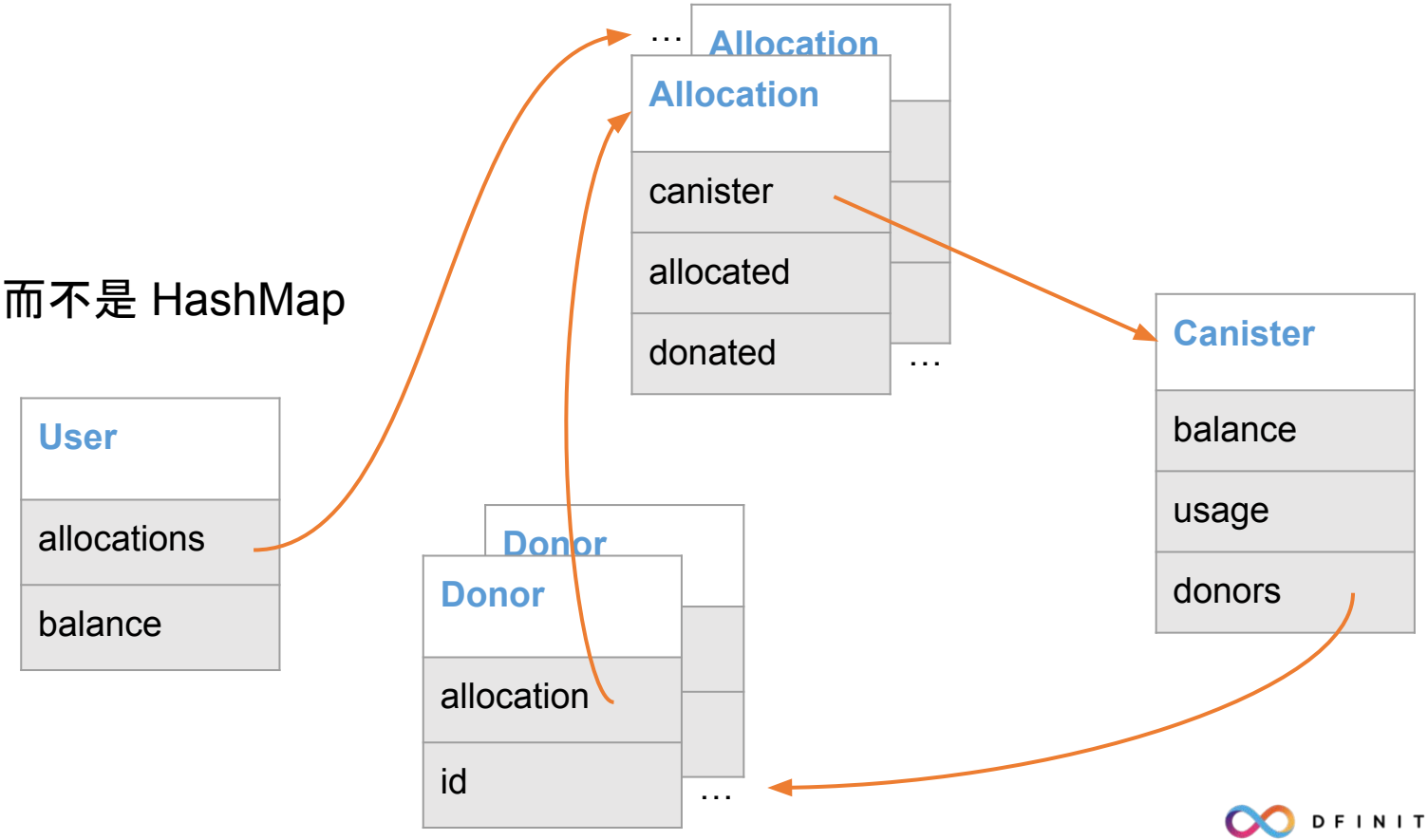


持久化数据结构

持久化两个表

- Users
- Canisters

采用了 Queue 而不是 HashMap



升级时改变数据结构

需求分析

- 保留原有的数据完整
- 支持数据结构的改动 (增加新字段、改变数据结构之间的关系)
- 是否使用 Stable Memory？

可选方案

- (模拟)关系数据库
- 在 postupgrade 里面处理从旧到新的转换

灾备方案

- 使用 logger 保持完整历史操作记录，必要时用来重建最新数据状态

Tip Jar 的公共接口

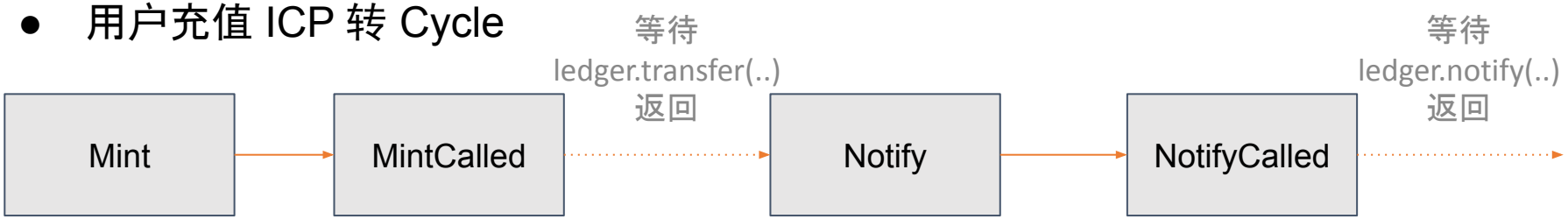
```
service {
  // 用户的账户信息
  aboutme: () -> (UserInfo) query;
  // 设置(或者调整)用户的 cycle 分配
  allocate: (AllocationInput) -> (variant {err: AllocationError; ok: UserInfo});
  // 用户转账后需要 ping 来完成 cycle 充值
  ping: (opt principal) -> () oneway;
  // 用户充值事务处理, called from this canister
  poll: () -> () oneway;
  // 容器充值事务处理, called from this canister
  topup: () -> () oneway;
};
```

事件处理的业务逻辑

两个 event queue，但一次只处理一件事务

先从队列中移除需要处理的事务对象，单独记录此事务的处理状态

- 用户充值 ICP 转 Cycle



- 给 Canister 补充 Cycle

异步事务的安全性

使用状态机，避免重入(reentrancy bug)

- 发出异步调用前先记录状态

注意状态的完整性和一致性

- 转账前，先从余额中扣除
- 转账如果失败，则归还到余额中

注意事务之间的相互影响

- 正确使用锁，确保释放，避免死锁

避免盲目升级破坏数据

- 可能有发出的调用尚未返回
- 所有发出的调用终将返回

前端的一些小技巧

缩短页面载入时间

1. 访问后端域名统一采用 ic0.app，避免域名查询的延迟
2. 先加载页面，推迟加载 agent, actor, etc.
3. 单页面应用，避免页面跳转

适配移动端界面

4. 选择合适的 CSS 框架
5. 适当隐藏一些元素

给用户足够的提示

6. 异步调用开始和结束的信号
7. 正确处理所有可能的错误，并反馈给用户

课程作业

实现一个简单的多人 Cycle 钱包：

1. 团队 N 个成员，每个人都可以用它控制和安装 canister。
2. 升级代码需要 M/N 成员同意。

要求：

- 前端管理 canister 的操作。
- 前端发起提案和投票的操作。
- 支持增加和删除小组成员的提案。