第六课

陈锡亮 Bryan
Polkadot Ambassador, Co-founder @ Laminar & Acala

bryan@laminar.one

内容

- 链上升级和数据迁移
- Substrate Kitty 设计和实现
- FRAME 资产相关模块介绍
  - assets
  - generic-asset
  - balances

链上升级和数据迁移

- Substrate 的 Runtime 是编译成 wasm 保存在链上状态之中
- 客户端从状态中得到 wasm runtime，然后使用执行器执行区块
- 无缝升级就是把新版的 wasm runtime 通过交易发送到链上，更新状态中的 wasm
- 流程：
  - 增加 spec_version
  - 把 runtime 编译成 wasm 文件
    - WASM_TARGET_DIRECTORY=$(pwd) cargo build
  - 通过治理模块进行升级
    - sudo.sudo(system.set_code(wasm))

链上升级和数据迁移

- Substrate 的链上状态可以理解为一个简单的 Key-Value Store
  - Storage Value
    - Twox128(module_prefix) ++ Twox128(storage_prefix)
  - Storage Map
    - Twox128(module_prefix) ++ Twox128(storage_prefix) ++ hasher(encode(key)
  - Storage Double Map
    - Twox128(module_prefix) ++ Twox128(storage_prefix) ++ hasher1(encode(key1)) ++ hasher2(encode(key2))

链上升级和数据迁移

- 任何状态数据结构的变化都需要进行数据迁移
  - Storage 重命名
  - 调整 hasher
  - 任何不是SCALE兼容的类型修改
- 数据升级和迁移的策略类似nosql数据库
  - 在一个block中进行升级全部状态
    - 只适用于少量的升级
  - Lazy upgrade
    - 只升级用到的数据

链上升级和数据迁移

- fn on_runtime_upgrade
  - 在升级后的block执行，优先于 on_initialize
- fn translate
  - 用旧的数据结构解码，然后转换成新的结构编码
- fn encode / fn decode
  - 手动实现编码解码处理版本兼容

Substrate Kitty 设计和实现

- 实现赠予小猫
  - fn transfer(origin, to: AccountId, kitty_id: u32)
- 目前的数据结构
  - OwnedKitties: map(AccountId, u32) => u32
  - OwnedKittiesCount: map AccountId => u32
- 复杂度
  - 随机删除用户的猫: O(n)
  - 添加新的猫到末尾: O(1)
- Swap & Pop 可以优化，但会修改顺序

Substrate Kitty 设计和实现

- 链表 LInked List
  - 随机删除数据: O(1)
  - 添加新的数据到末尾: O(1)
  - 随机位置添加数据: O(n)
- 实现
  - struct KittyLinkedItem
    - prev: Option<u32>
    - next: Option<u32>
  - OwnedKitties: map(AccountId, Option<u32> => Option<KittyLinkedItem>

Substrate Kitty 设计和实现

- 双向遍历
  - head.next -> item1.next -> item2.next -> None
  - head.prev -> item2.prev -> item1.prev -> None
- 空表: []
  - head: { prev: head, next: head }
- 添加元素 item1: [item1]
  - head: { prev: item1, next: item1 }
  - item1: { prev: head, next: head }

Substrate Kitty 设计和实现

- 添加元素 item2: [item1, item2]
  - head: { prev: item2, next: item1 }
  - item1: { prev: head, next: item2 }
  - item2: { prev: item1, next: head }
- 添加元素 item3: [item1, item2, item3]
  - head: { prev: item3, next: item1 }
  - item1: { prev: head, next: item2 }
  - item2: { prev: item1, next: item3 }
  - item3: { prev: item2, next: head }

Substrate Kitty 设计和实现

- 删除元素 item2: [item1, item3]
  - head: { prev: item3, next: item1 }
  - item1: { prev: head, next: item3 }
  - item3: { prev: item1, next: head }
- 删除元素 item3: [item1]
  - head: { prev: item1, next: item1 }
  - item1: { prev: head, next: head }
- 删除元素 item1: [ ]
  - head: { prev: head, next: head }

FRAME 资产相关模块介绍

- pallet-assets
  - 非常简单的多资产管理模块
  - 不建议生产环境使用
- pallet-balances
  - 单一资产管理模块
  - 可以通过多个 instance 来实现多资产管理
- pallet-generic-asset
  - 多资产管理模块
  - 实现了 trait Currency，可以和其他的模块兼容

作业

1. 补完剩下的代码 https://github.com/SubstrateCourse/substrate-kitties/blob/lesson6/pallets/kitties/src/lib.rs
2. 对比 pallet-asset 和 pallet-balances，简单分析下 pallet-asset 都有哪些缺失的使得其不适合生产环境使用。
   a. 注：以太坊的问题之一是状态爆炸
3. 简单的分析下为什么 Polkadot 配置的 Balance 类型是 u128，而不是类似以太坊的 u256。
   a. 注：DOT 发行量是 1000 万个，精度是12位，年增发率是 10%