Substrate 区块链应用开发

交易费用和权重

孙凯超
kaichao@parity.io

获取帮助: https://substrate.io

内容

- 交易费用的存在原因
- 设计思路
- 组成部分
- FRAME 模块导读

存在原因 - Web 2.0

服务提供方利用用户的个人信息、产生的数据、注意力等来变现：

- 广告推送；
- 用户数据分析指导商家决策；
- 直接共享、贩卖用户隐私等。

** 用户数据的所有者是服务提供方

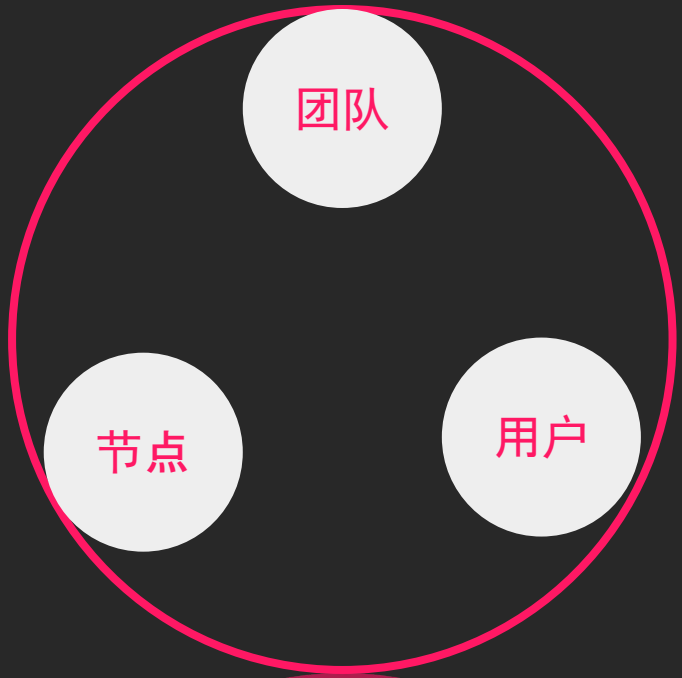
存在原因 - Web 3.0

用户通过私钥掌握数据，敏感数据可以通过加密防止被窃取。

存在原因 - Web 3.0

天下没有免费的午餐，享受自由也要支付服务费用，用来：

- 激励参与方更加有效的协作；
- 调节资源的利用率



如何设计交易费用

典型的资源限制：

- 不同地区带宽差异巨大
- 有限的区块大小
- 有限的区块生成时间
- 昂贵的链上存储空间
- 如何分配交易费用

如何设计交易费用

典型的资源限制：

- 不同地区带宽差异巨大
- 有限的区块大小
- 有限的区块生成时间
- 昂贵的链上存储空间
- 如何分配交易费用

可用的设计思路：

- 计算每笔交易占用的字节数来收取费用
- 计算或者性能测试得出不同交易所消耗的时间
- 一次性付费和租赁两种模式
- 通过链上治理进行分配

Substrate 交易费用组成

总费用 = 基本费用 +(字节费用 + 权重费用)*(1 + 动态调节费率)+ 小费

Substrate 交易费用组成

总费用 = 基本费用 +(字节费用 + 权重费用)*(1 + 动态调节费率)+ 小费

参数:

- ExtrinsicBaseWeight
  125 * WEIGHT_PER_MICROS
- MaximumBlockWeight
  2 * WEIGHT_PER_SECOND
- WeightToFee

字节费用

总费用 = 基本费用 +(字节费用 + 权重费用)*(1 + 动态调节费率)+ 小费

参数和特点：

- 字节数是SCALE编码后的长度
- 最大区块长度
  MaximumBlockLength
- 每字节的费用
  TransactionByteFee
- 配置在可升级的runtime代码

字节费用 = 每字节费用 * 字节数

字节费用

总费用 = 基本费用 +(字节费用 + 权重费用)*(1 + 动态调节费率)+ 小费

参数和特点：

- 字节数是SCALE编码后的长度
- 最大区块长度
  MaximumBlockLength
- 每字节的费用
  TransactionByteFee
- 配置在可升级的runtime代码

字节费用 = 每字节费用 * 字节数

** Kusama网络：5MB，0.0001ksm

权重费用

总费用 = 基本费用 +(字节费用 + 权重费用)*(1 + 动态调节费率)+ 小费

权重被用来定义交易产生的计算复杂度:

- 区块的总权重 MaximumBlockWeight
- 两种不同级别的交易类型
  - Normal
  - Operational
- 可用区块比 AvailableBlockRatio
- 三种设置权重的方式:固定，自定义，FunctionOf

** Kusama网络:区块的总权重 2,000,000,000,000;可用区块比:75%

固定权重

```
// 固定权重的Normal交易
#[weight = 10_000]
fn accumulate_dummy(origin, increase_by: T::Balance) -> DispatchResult {
	// --snip--
}

// 固定权重的Operational交易
#[weight = (2_000_000, Operational)]
fn accumulate_dummy(origin, increase_by: T::Balance) -> DispatchResult {
	// --snip--
}
```

自定义权重

自定义权重计算方法，自定义的结构体，并实现接口：

- WeighData，计算权重
- ClassifyDispatch，判断交易级别
- PaysFee，设置是否付费标志位

自定义权重

```
// The struct with a multiplier
struct WeightForSetDummy<T:  pallet_balances::Trait>(BalanceOf<T>);

/// A type alias for the balance type.
type BalanceOf<T> = <T as pallet_balances::Trait>::Balance;

impl<T:  pallet_balances::Trait>  WeighData<(&BalanceOf<T>,)> for WeightForSetDummy<T>
{
        fn weigh_data(&self, target: (&BalanceOf<T>,)) -> Weight {
                let multiplier = self.0;
                (*target.0 * multiplier).saturated_into::<Weight>()
        }
}
```

自定义权重

```
impl<T: pallet_balances::Trait> ClassifyDispatch<(&BalanceOf<T>,)> for WeightForSetDummy<T> {
	fn classify_dispatch(&self, target: (&BalanceOf<T>,)) -> DispatchClass {
		if *target.0 > <BalanceOf<T>>::from(1000u32) {
			DispatchClass::Operational
		} else {
			DispatchClass::Normal
		}
	}
}

impl<T: pallet_balances::Trait> PaysFee<(&BalanceOf<T>,)> for WeightForSetDummy<T> {
	fn pays_fee(&self, _target: (&BalanceOf<T>,)) -> Pays {
		Pays::Yes
	}
}
```

自定义权重

如何使用:

```
// Apply weight to dispatchable call
#[weight = WeightForSetDummy::<T>(<BalanceOf<T>>::from(100u32))]
fn set_dummy(origin, #[compact] new_value: T::Balance) {
        // --snip--
}}
```

FunctionOf 结构体

FunctionOf 接收三个参数:

- 权重值或根据参数计算权重的closure
- 固定交易级别或计算交易级别的closure
- 是否付费的标志位或closure

FunctionOf 结构体

```
// weight = a x 10 + b
#[weight = FunctionOf(|args: (&u32, &u32)| args.0 * 10 + args.1, DispatchClass::Normal, Pays::Yes)]
fn f1(_origin, _a: u32, _b: u32) { unimplemented!(); }

#[weight = FunctionOf(|_: (&u32, &u32)| 0, DispatchClass::Operational, Pays::Yes)]
fn f2(_origin, _a: u32, _b: u32) { unimplemented!(); }
```

权重费用

总费用 = 基本费用 +(字节费用 + 权重费用)*(1 + 动态调节费率)+ 小费

权重被用来定义交易产生的计算复杂度:

- 合理的权重值需要通过性能测试来获取
- 可调用函数的注释中要给出计算复杂度和数据读写操作
- 通过WeightToFee, 转换权重值为权重费用

权重费用

```
pub struct WeightToFee;

impl WeightToFeePolynomial for WeightToFee {
    type Balance = Balance;
    fn polynomial() -> WeightToFeeCoefficients<Self::Balance> {
        // -- snip --
    }
}
```

动态调节费率

总费用 = 基本费用 +(字节费用 + 权重费用)*(1 + 动态调节费率)+ 小费

网络平稳运行的过程中，区块资源的使用比例应该稳定：

- TargetBlockFullness参数，通常为25%
- 当前区块资源使用超过25%时，将下一区块动态调节费率设置为正，增加交易费用；
- 当资源使用率不足25%时，将下一区块的动态调节费率设置为负，减少交易费用，鼓励交易的发生

小费

总费用 = 基本费用 +(字节费用 + 权重费用)*(1 + 动态调节费率)+ 小费

不是必须的，具体数量由交易发送者决定，并且完全由区块生产者获得；

而交易费用的其它组成部分会根据一定的比例分配进入“国库”。

资料

代码:

https://github.com/paritytech/substrate/tree/master/frame/example

https://github.com/kaichaosun/play-substrate/tree/master/pallets/weight

文档:

https://substrate.dev/docs/en/development/module/fees

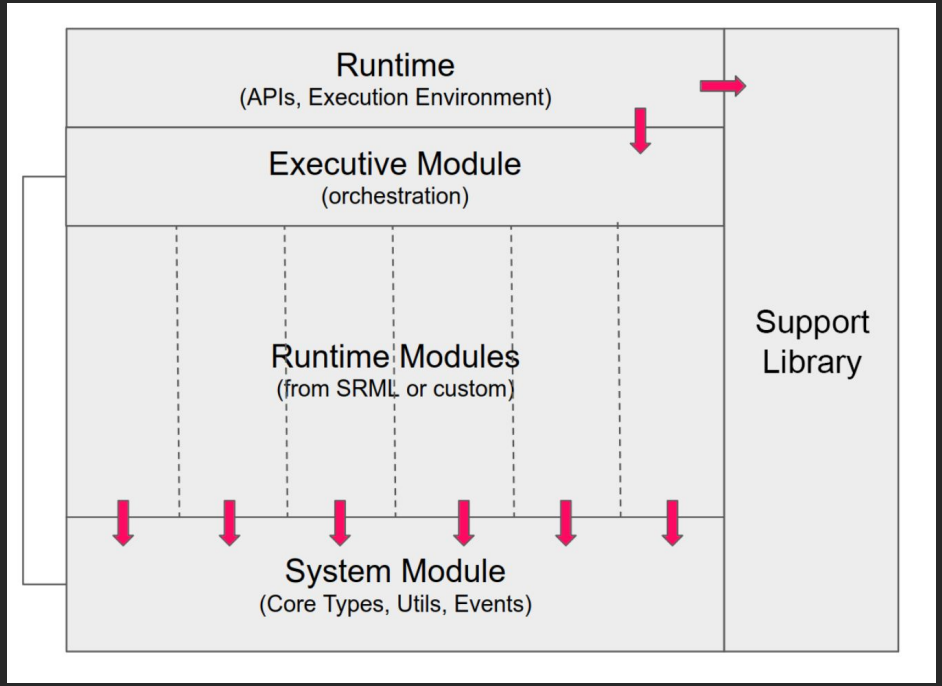
https://zhuanlan.zhihu.com/p/108194544

FRAME 模块导读

- system
- timestamp
- transaction-payment
- utility

FRAME 组成

- System: 底层 types, storages, functions
- Support：宏, traits
- Executive：runtime编排层
- Pallets：功能模块



Questions?

官网文档：substrate.io

知乎专栏：parity.link/zhihu

kaichao@parity.io

weixin415148673