学会 Substrate 区块链应用开发

8. Off-chain Worker 教程 I

Jimmy Chu
jimmy.chu@parity.io

获取帮助: **https://substrate.io**

内容

- 什么是 off-chain worker
- ocw 能做什么？
- ocw 的运作原理
- 代码时间
  - 触发 ocw
  - 作具签名交易
  - 作不具签名交易
  - 单元测试
- 资源

什么是 off-chain worker？

或作: 链下工作机, ocw

一般区块链開發的制约

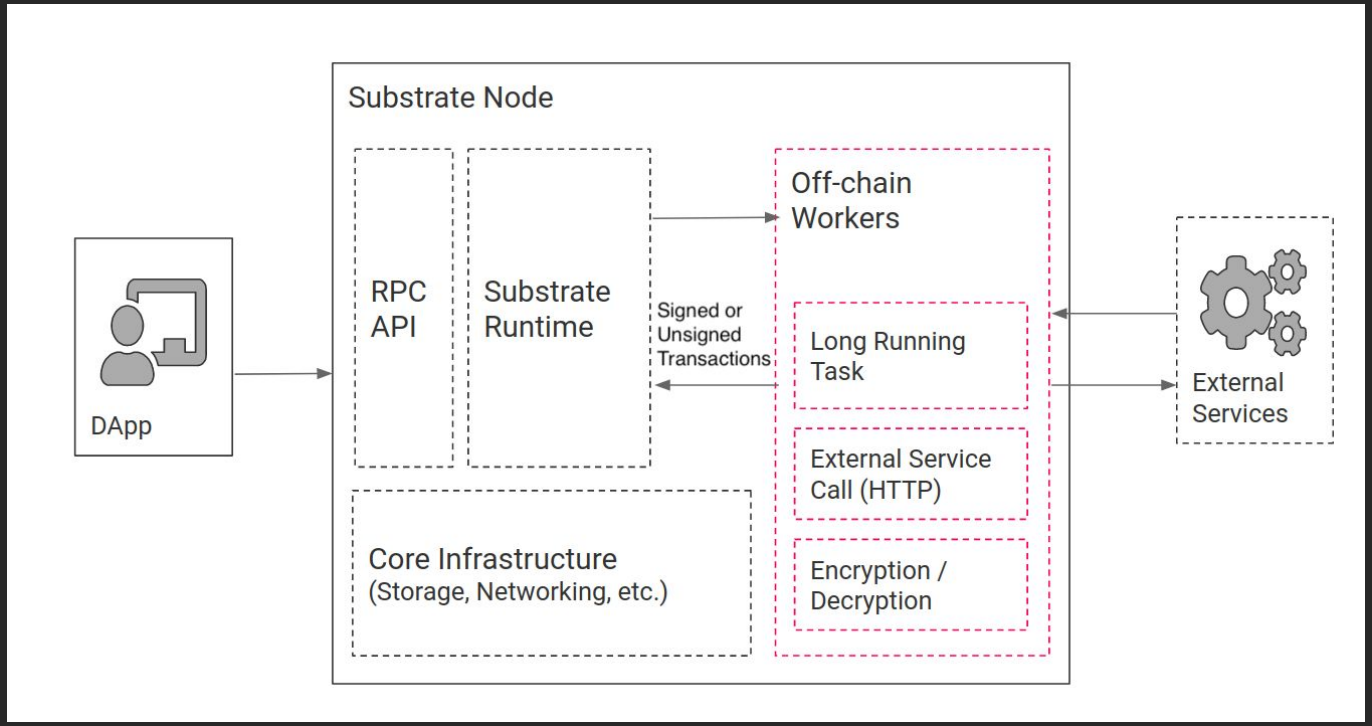
- 计算量不能太大
  - 计算量太大，需时太长会影响出块时间
- 不能做具不确定结果的操作，简单如从外部取数据也做不了
  - 不确定返回结果 (结果不确定，整条链则不能达成共识)
  - 不确定返回时间

ocw 也因应而生

它可以:

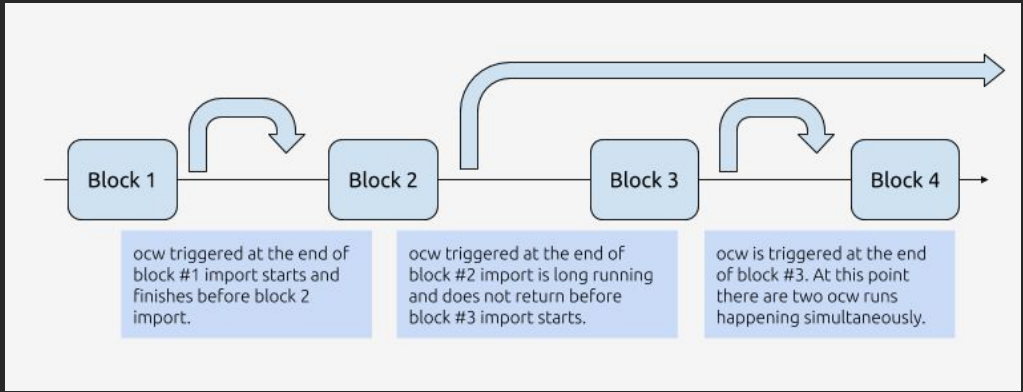
- ocw 做的操作不需要共识
- 当模块拥有独立密钥对时，可对交易进行签名并提交 (这课)
- 可提交不具签名交易 (这课)
- 有一个功能齐全的 HTTP 客户端库，能访问外部服务并获取数据 (下课)
- 有一个私有的数据库，供 ocw 储存数据 (下课)

ocw 运作原理



ocw 运作原理

- 在 runtime 进程以外运行，不影响出块，兼能读取 runtime 存储
- 在每一区块完成导入后执行一次
- ocw 进程可以在下一个区块进程启动前完成，也可横跨多个区块导入时期。所以同一时间可以有多于一个 ocw在运行，由不同区块来触发



接下来开始代码时间 🙂

准备

Substrate Recipe 里的 offchain-demo pallet

https://github.com/substrate-developer-hub/recipes/tree/master/pallets/offchain-demo

触发 ocw

- 在 `decl_module! {}` 内加一个 `fn offchain_worker()` 的函数
- 每次在区块导入后，链下工作机就会回调这函数，也是使用链下工作机的起始入口

```
decl_module! {
  // ...
  fn offchain_worker(block_number:
T::BlockNumber) {
    debug::info!("Entering off-chain
workers");
    // 你定义的链下处理逻辑
  }
}
```

在 ocw 内作具签名交易

1. 导入适当的库，设定 Pallet 的 Trait

```
use frame_system::{
  offchain::{
    AppCrypto, CreateSignedTransaction, SendSignedTransaction, Signer,
SubmitTransaction,
  },
};


/// This is the pallet's configuration trait
pub trait Trait: system::Trait + CreateSignedTransaction<Call<Self>> {

  /// The identifier type for an offchain worker.
  type AuthorityId: AppCrypto<Self::Public, Self::Signature>;
  ...
}
```

在 ocw 内作具签名交易

2. 調用這 pallet 的 runtime 要实现 3 个特征：

- frame_system::offchain::CreateSignedTransaction
- frame_system::offchain::SigningTypes
- frame_system::offchain::SendTransactionTypes

```
impl<LocalCall>
frame_system::offchain::CreateSignedTransaction<LocalCall>
for Runtime
where
  Call: From<LocalCall>,
{
  fn create_transaction<C:
frame_system::offchain::AppCrypto<Self::Public,
Self::Signature>>(
    call: Call,
    public: <Signature as
sp_runtime::traits::Verify>::Signer,
    account: AccountId,
    index: Index,
  ) -> Option<(
    Call,
    <UncheckedExtrinsic as
sp_runtime::traits::Extrinsic>::SignaturePayload,
  )> {
    //...
  }
}
```

parity 一块+ OneBlock+

在 ocw 内作具签名交易

```
impl frame_system::offchain::SigningTypes for Runtime {
  type Public = <Signature as sp_runtime::traits::Verify>::Signer;
  type Signature = Signature;
}

impl<C> frame_system::offchain::SendTransactionTypes<C> for Runtime
where
  Call: From<C>,
{
  type OverarchingCall = Call;
  type Extrinsic = UncheckedExtrinsic;
}
```

在 ocw 内作具签名交易

```
// 在 pallet lib.rs 里
pub mod crypto {
  use crate::KEY_TYPE;
  use sp_core::sr25519::Signature as
Sr25519Signature;
  use sp_runtime::{
    app_crypto::{app_crypto, sr25519},
    traits::Verify,
    MultiSignature, MultiSigner,
  };

  app_crypto!(sr25519, KEY_TYPE);

  pub struct TestAuthId;
  // implemented for ocw-runtime
  impl
frame_system::offchain::AppCrypto<MultiSigner,
MultiSignature> for TestAuthId {
    type RuntimeAppPublic = Public;
    type GenericSignature =
sp_core::sr25519::Signature;
    type GenericPublic = sp_core::sr25519::Public;
  }
}
```

3. 定义这 pallet 用作签名的密钥对

最后在 runtime 里设置 AuthorityId

```
impl offchain_demo::Trait for Runtime {
  type AuthorityId =
offchain_demo::crypto::TestAuthId;
  type Call = Call;
  type Event = Event;
}
```

parity 一块+ OneBlock+

在 ocw 内作具签名交易

4.1 取得 Signer

4.2 用 Signer 调用 send_signed_transaction

4.3 查看提交交易结果

```
fn signed_submit_number(block_number: T::BlockNumber) ->
Result<(), Error<T>> {
  // 2.1 取得 Signer
  let signer = Signer::<T, T::AuthorityId>::all_accounts();

  // 2.2 用 Signer 调用 send_signed_transaction
  let results = signer.send_signed_transaction(|_acct| {
    // We are just submitting the current block number back
on-chain
    Call::submit_number_signed(submission)
  });

  // 2.3 查看提交交易结果
  for (acc, res) in &results {
    match res {
      Ok(()) => debug::native::info!("success"),
      Err(e) => debug::error!("error")
    };
  }
}
```

在 ocw 内作不具签名交易

- 在 pallet 实现 frame_support::unsigned::ValidateUnsigned
- 并把可接受不具签名交易的函数定义在这允许名单内

```
use sp_runtime::transaction_validity::{
  InvalidTransaction, TransactionPriority,
TransactionSource, TransactionValidity, ValidTransaction,
};


fn validate_unsigned(_source: TransactionSource, call:
&Self::Call) -> TransactionValidity {
  #[allow(unused_variables)]
  if let Call::submit_number_unsigned(number) = call {
    debug::native::info!("off-chain send_unsigned:
number: {}", number);

    ValidTransaction::with_tag_prefix("offchain-demo")
      .priority(T::UnsignedPriority::get())
      .and_provides([b"submit_number_unsigned"])
      .longevity(3)
      .propagate(true)
      .build()
  } else {
    InvalidTransaction::Call.into()
  }
}
```

在 ocw 内作不具签名交易

用 `SubmitTransaction` 来提交不具签名交易

```
let call = Call::submit_number_unsigned(submission);

SubmitTransaction::<T,
Call<T>>::submit_unsigned_transaction(call.into()).map_err(
|e| {
  debug::error!("Failed in unsigned_submit_number: {:?}",
e);
  <Error<T>>::UnsignedSubmitNumberError
})
```

在 ocw 内作不具签名交易

在 Runtime 内, 在 construct_runtime! 把 `ValidateUnsigned` 传到 pallet enum 内

```
construct_runtime!(
  pub enum Runtime where
    Block = Block,
    NodeBlock = opaque::Block,
    UncheckedExtrinsic = UncheckedExtrinsic
  {
    ...
    // The Recipe Pallets
    OffchainDemo: offchain_demo::{Module, Call, Storage,
Event<T>, ValidateUnsigned},
  }
);
```

对 ocw 作单元测试

两个重点

- 建立一个 mock 的 runtime：这个 mock 和没有 ocw 时的有大不同，建议可参考 offchain-demo 里的代码。

- 测试 ocw, 是测试 ocw 所变更了的状态，比如交易池是否多了一条交易。而不要测试该交易内的逻辑

```
#[test]
fn offchain_send_signed_tx() {
  let (mut t, pool_state, offchain_state) =
ExtBuilder::build();

  t.execute_with(|| {
    // when
    let num = 32;
    OffchainDemo::send_signed(num).unwrap();

    // Test only one transaction is in the pool.
    let tx =
pool_state.write().transactions.pop().unwrap();

assert!(pool_state.read().transactions.is_empty());

    // Test the transaction is signed
    let tx = TestExtrinsic::decode(&mut
&*tx).unwrap();
    assert_eq!(tx.signature.unwrap().0, 0);

    // Test the transaction is calling the expected
extrinsics with expected parameters
    assert_eq!(tx.call,
Call::submit_number_signed(num));
  });
}
```

parity 一块+ OneBlock+

资源

- Substrate knowledge base
  https://substrate.dev/docs/en/knowledgebase/learn-substrate/off-chain-workers
- Substrate Recipe - offchain-demo
  https://substrate.dev/recipes/3-entrees/off-chain-workers/index.html
- 知乎专栏-链下工作机系列
  https://zhuanlan.zhihu.com/p/137428214

Questions?

官网文档：substrate.dev

知乎专栏：parity.link/zhihu

jimmy.chu@parity.io