Moonbeam

Moonbeam开发进阶课程

批处理预编译

PureStake开发者关系团队

课程导航

批处理预编译概述

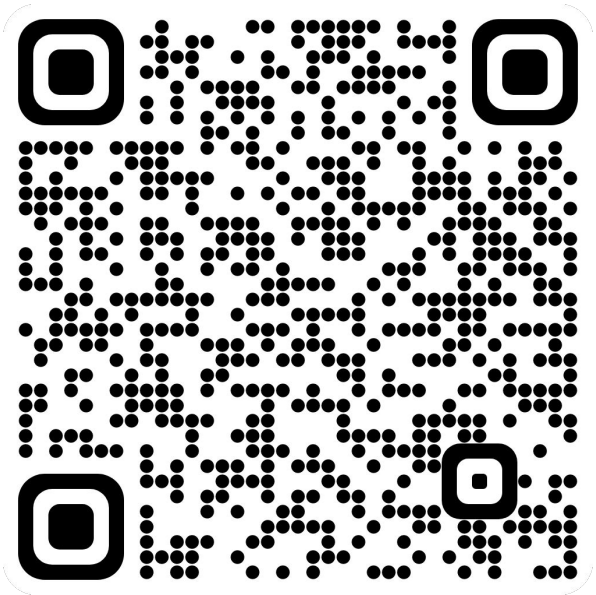
批处理转帐

批处理调用合约

获取CallData

课程资源GitHub Repo

https://github.com/PureStake/moonbuilders-academy/tree/main/chinese/advanced-course



批处理预编译概述

什么是预编译？

- 一段预先编译的代码或智能合约
- 最初被以太坊所使用，用于常用的加密和哈希算法，如SHA256、RIPEMD56、Keccak256等
- 一种Substrate原语，是构建跨链交互和与Substrate pallet交互的重要组成部分
- 预编译方法一般会跳过 EVM 执行，而是直接在本地处理节点运行时上计算
- 更详细预编译介绍：https://academy.china.moonbeam.network/courses/moonbeam-intro-course/lectures/40556765
- 完整Moonbeam预编译列表：https://github.com/PureStake/moonbeam/tree/master/precompiles

批处理(Batch)预编译

- 允许用户将多个子调用合并为一笔交易，即同时签署多笔交易；可以改善用户体验和优化Gas消耗
- 提供三种执行模式：BatchSome, BatchSomeUntilFailure, BatchAll
- 预编译地址为：0x0000000000000000000000000000000000000808

批处理预编译接口定义

方法：

- **batchSome(*address[]* to, *uint256[]* value, *bytes[]* call_data, *uint64[]* gas_limit)** — 执行多个调用，其中每个数组的相同索引合并到单个子调用所需的信息。如果某一子调用回滚状态，仍将尝试执行其余子调用
- **batchSomeUntilFailure(*address[]* to, *uint256[]* value, *bytes[]* call_data, *uint64[]* gas_limit)** — 执行多次调用，其中每个数组的相同索引合并到单个子调用所需的信息。如果某一子调用回滚状态，则不会尝试执行后续子调用
- **batchAll(*address[]* to, *uint256[]* value, *bytes[]* call_data, *uint64[]* gas_limit)** — 以原子方式执行多个子调用，其中每个数组的相同索引组合成单个子调用所需的信息。如果任何子调用执行失败，所有子调用都将回滚状态

事件：

- **SubcallSucceeded(*uint256* index)** - 当给定索引的子调用成功时发出
- **SubcallFailed(*uint256* index)** - 当给定索引的子调用失败时发出

Batch.sol

https://github.com/PureStake/moonbeam/blob/master/precompiles/batch/Batch.sol

通过Remix IDE交互预编译

批处理转帐

批处理调用合约

xcUNITBridge示例(2)

使用批处理预编译优化用户体验

批量两个操作：

1. 在xcUNIT ERC20合约上授权 X 数额
2. 在xcUNITBridge合约上调用send_tokens方法

批量处理”batchAll”调用参数：

to: ["0xFfFFfFff1FcaCBd218EDc0EbA20Fc2308C778080", "0x5B55F89d4fDe76de4D4b96A1618A67c24D9872Ad"]
value: [0, 0]
callData:
["0x095ea7b3000000000000000000000000005b55f89d4fde76de4d4b96a1618a67c24d9872ad0000000000000000000000000000000000000000000000000000000000000000174876e800",
"0x41a68f8d0000000000000000000000000000000000000000000000000000000000000040000000000000000000000000000000000000000000000000000000000000000174876e8000000000000000000000000000000000000000000000000000000000000000100000000000000000000000000000000000000000000000000000000000000400000000000000000000000000000000000000000000000000000000000000001000000000000000000000000000000000000000000000000000000000000000200000000000000000000000000000000000000000000000000000000000000022019c6af76cd6513e44e421ab8dc8f9d86e102200eb2e55cbc41a2d5ce214ee254f0000000000000000000000000000000000000000000000000000000000000000"]
gasLimit: []

通过以太坊库获取CallData

web3.js

```
// Import the contract ABI
const { abi } = require('./YOUR-ABI-PATH');

// Find call data for the setMessage function
const callData =
web3.eth.abi.encodeFunctionCall(
  abi
  [
    "INPUT-1-HERE",
    "INPUT-2-HERE",
    ...
  ]
);
```

ethers.js

```
// Import the contract ABI
const { abi } = require('./YOUR-ABI-PATH');

// Use ABI to create an interface
const yourContractInterface = new
ethers.utils.Interface(abi);

// Find call data for the setMessage function
const callData =
yourContractInterface.encodeFunctionData(
  'FUNCTION-NAME-HERE',
  [
    "INPUT-1-HERE",
    "INPUT-2-HERE",
    ...
  ]
);
```