MTSC2020

中国互联网测试开发大会·深圳站

TESTING SUMMIT CONFERENCE CHINA 2020

2020.11.20-21 | 中国·深圳宝立方国际酒店

主办方：TesterHome

智能合约测试体系建设及实践

刘 超
腾讯支付与金融线　高级专项测试工程师

- PART 01

  区块链智能合约介绍

- PART 02

  智能合约多维度测试体系

- PART 03

  测试实践案例

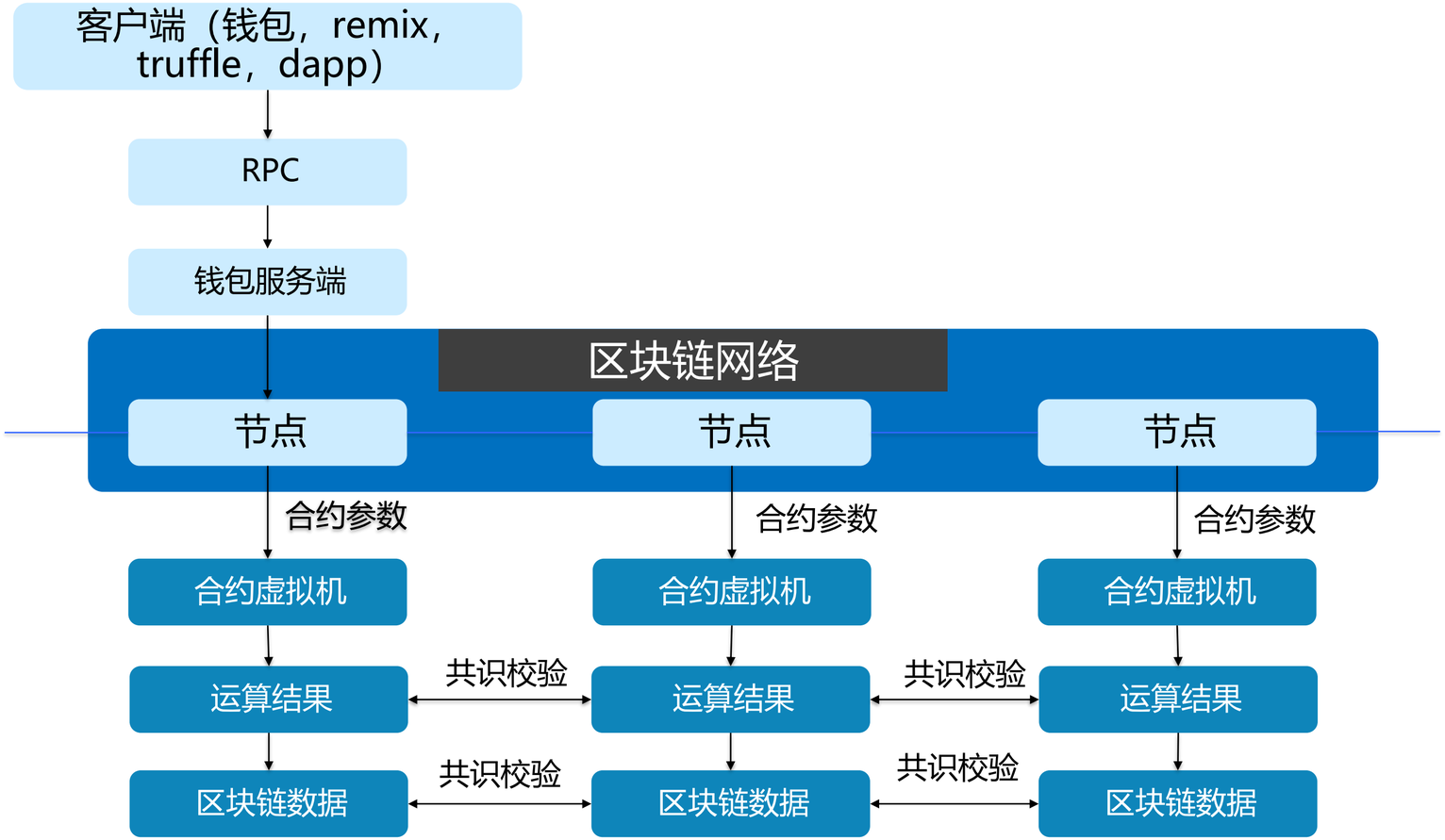
智能合约介绍

数字承诺

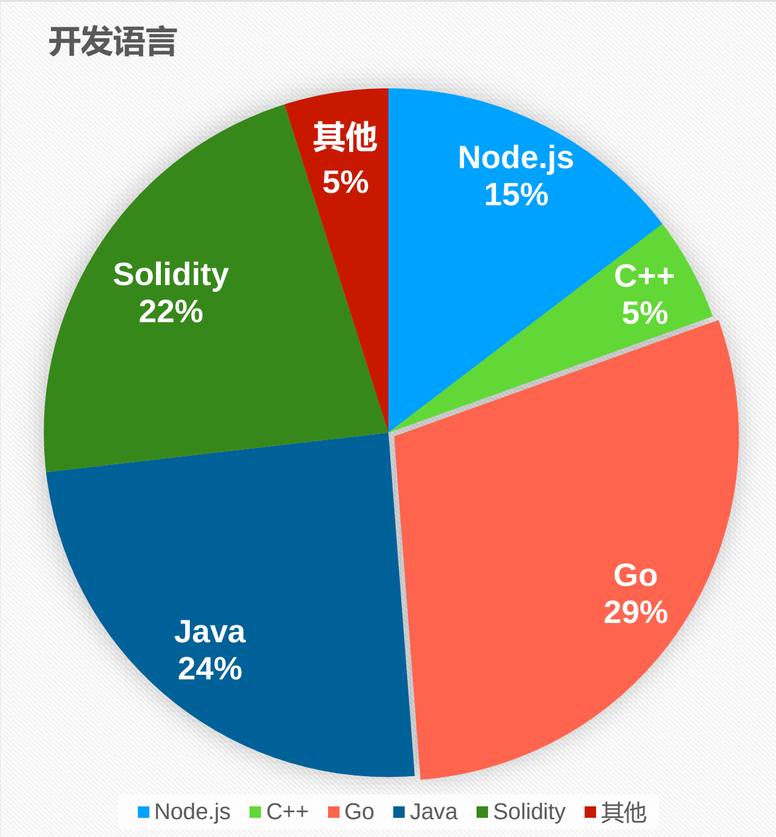
- 代码实现
- 达成条件执行

链上应用

- 去中心化
- 不可篡改



智能合约开发语言



数据来源：TBI测试

```
pragma solidity >=0.4.22 <0.7.0;

/**
 * @title Ballot
 * @dev Implements voting process along with vote delegation
 */
contract Ballot {

    struct Voter {
        uint weight; // weight is accumulated by delegation
        bool voted;  // if true, that person already voted
        address delegate; // person delegated to
        uint vote;   // index of the voted proposal
    }

    struct Proposal {
        // If you can limit the length to a certain number of bytes,
        // always use one of bytes1 to bytes32 because they are much cheaper
        bytes32 name;   // short name (up to 32 bytes)
        uint voteCount; // number of accumulated votes
    }

    address public chairperson;

    mapping(address => Voter) public voters;

    Proposal[] public proposals;

    /**
     * @dev Create a new ballot to choose one of 'proposalNames'.
     * @param proposalNames names of proposals
     */
    constructor(bytes32[] memory proposalNames) public {
        chairperson = msg.sender;
        voters[chairperson].weight = 1;

        for (uint i = 0; i < proposalNames.length; i++) {
            // 'Proposal({...})' creates a temporary
            // Proposal object and 'proposals.push(...)'
            // appends it to the end of 'proposals'.
            proposals.push(Proposal({
                name: proposalNames[i],
                voteCount: 0
            }));
        }
    }
```

版本声明

合约主体

合约变量

合约方法

Solidity智能合约示例

- PART 01

  区块链智能合约介绍

- PART 02

  智能合约测试维度

- PART 03

  测试方法与实践案例

智能合约与传统业务区别

	传统业务	智能合约
提测物	二进制文件	源代码
运行环境	操作系统	区块链
访问地址	Server(ip:port)	链上地址
调用方式	接口调用	区块链交易
迭代方式	替换升级	全新发布
回退方式	可回退	无法回退
数据变更	修改数据库	经过全网共识
测试手段	主要是系统测试	大多数是单元测试
执行成本	物理费用、流量等	消耗GAS

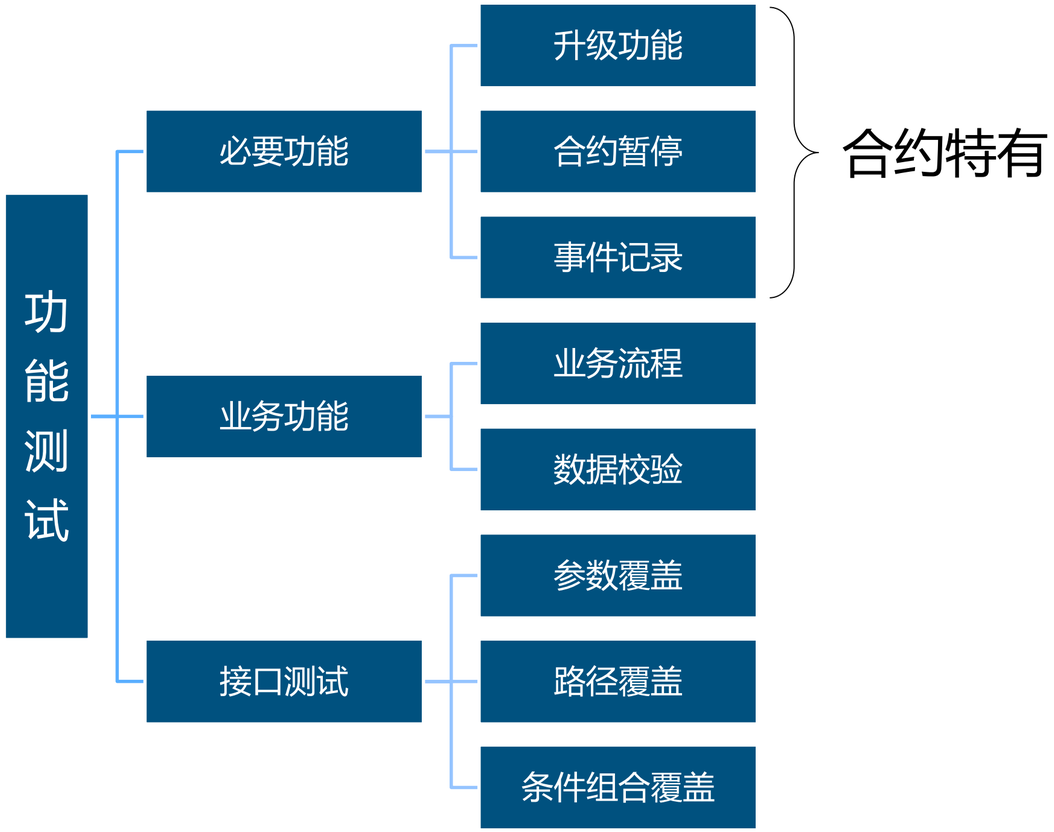
智能合约测试维度

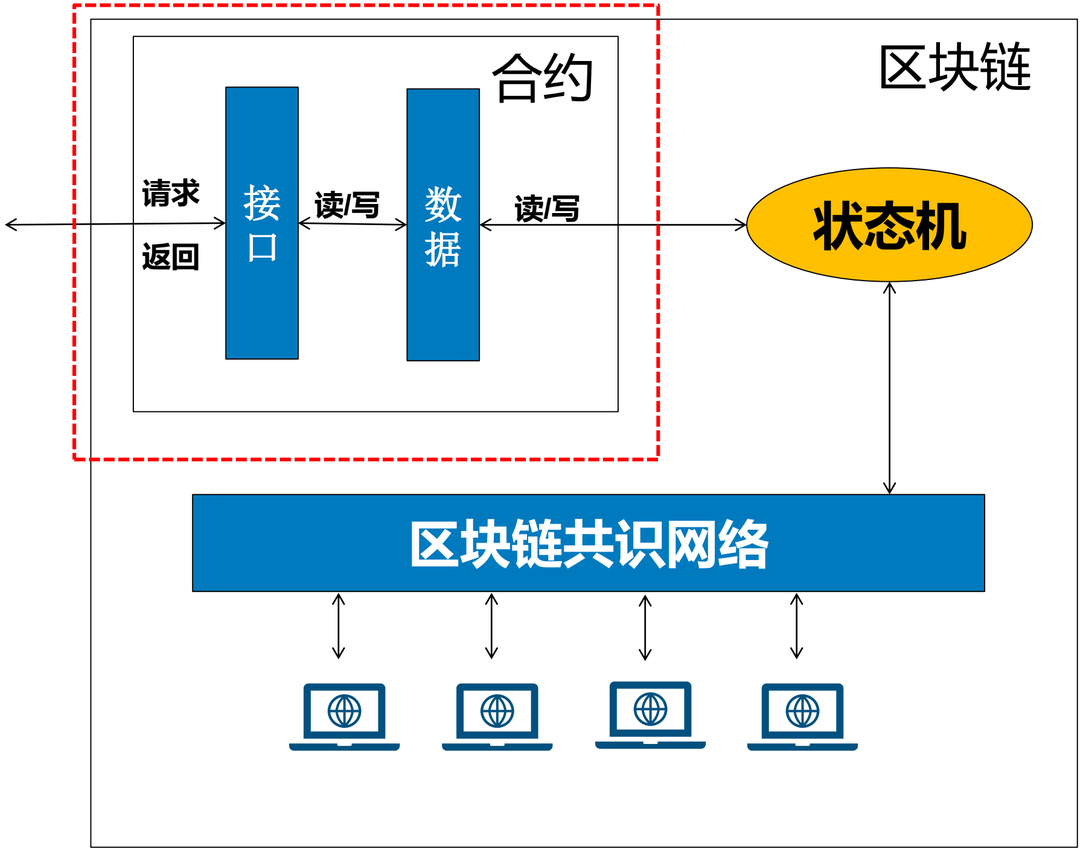
功能测试

安全验证

成本审计

智能合约功能测试

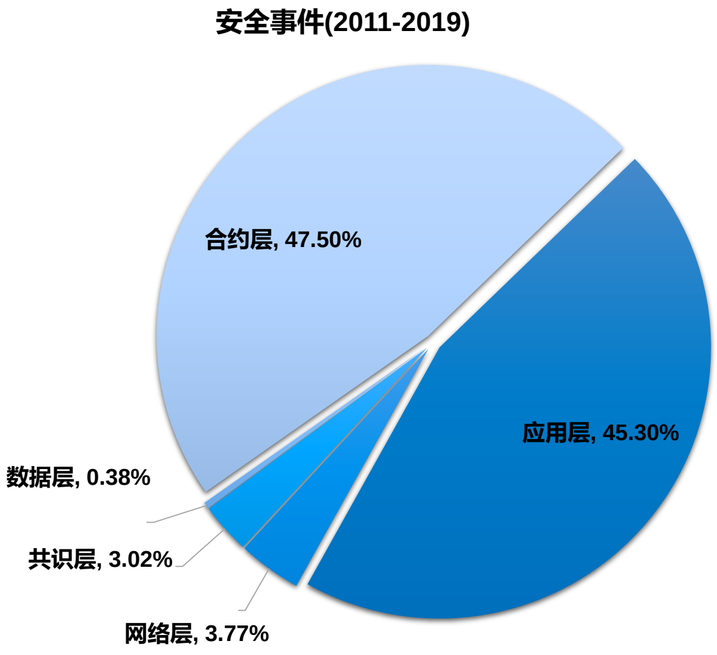

功能测试
必要功能
业务功能
接口测试
升级功能
合约暂停
事件记录
业务流程
数据校验
参数覆盖
路径覆盖
条件组合覆盖
合约特有


区块链
合约
请求
返回
接口
读/写
数据
读/写
状态机
区块链共识网络

智能合约安全测试

安全审计	• 第三方机构审计 • 常见安全漏洞检查 ←	
合约加密	• 源码无法明文读取 • 源码避免开源	开发规范
规范设计	• 开发合约模板 • 私钥安全保存	开发规范
升级恢复	• 应急暂停 ✓ • 升级合约	运维功能
形式化验证	• 数学建模 ✓ • 系统性推理	

中国信通院《区块链安全白皮书(2018)》



数据来源：白帽汇安全研究院

智能合约成本审计

- **什么是Gas?**
  - 计算成本
  - 存储成本
  - 激励矿工
- **如何减少gas消耗?**
  - 编译优化部署
  - 减少数据规模
  - 降低复杂度
- **目标:**
  - 减少运营执行合约成本

Transaction details

0x784a45639003e57d9936f246ecd6b9eb7889d6ae5687e574938432433ef6faba

Hash	0x784a45639003e57d9936f246ecd6b9eb7889d6ae5687e574938432433ef6faba
Block	10588255 | 1 Confirmation
Time:	2020年8月4日 01:48:17 (a minute ago)
From	0x4c39ADA0340c1Eb3CeE343F44819323dD29081A9
To	0xECc54876755fE3E04eeb7B23AE4d3d4b0302bbB8
Value	0 ETH | 0 USD
Fee	0.05496 ETH | $21.47
Gas Price	260 GWei
Gas Limit	2,000,000
Gas Used	211,366
Actions	↳ Invoked 0x84cA8bc799727... sending 0 ETH consuming 38072 Gas of 1929038
	↳ Invoked 0x84cA8bc799727... sending 0 ETH consuming 22565 Gas of 1829410

- PART 01

  区块链智能合约介绍

- PART 02

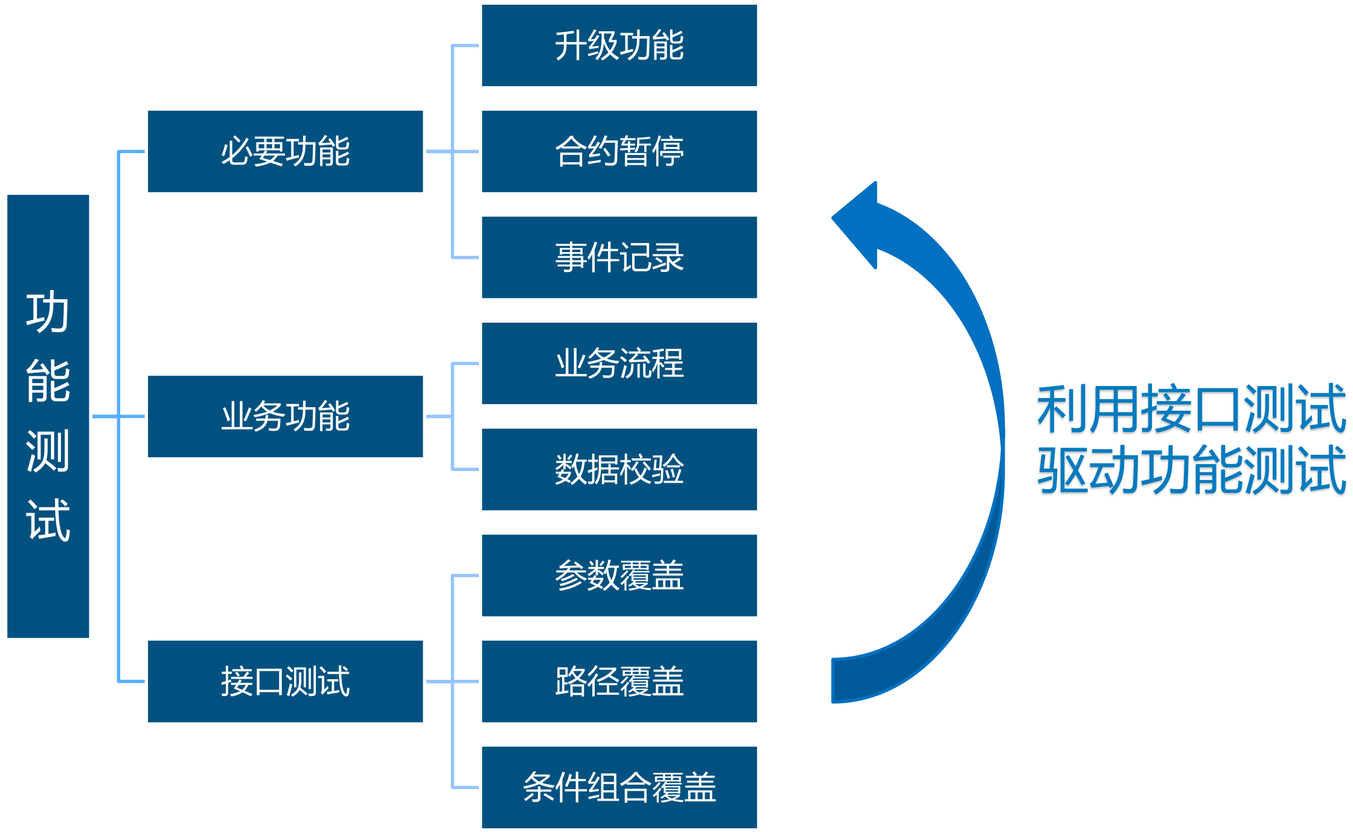
  智能合约测试维度

- PART 03

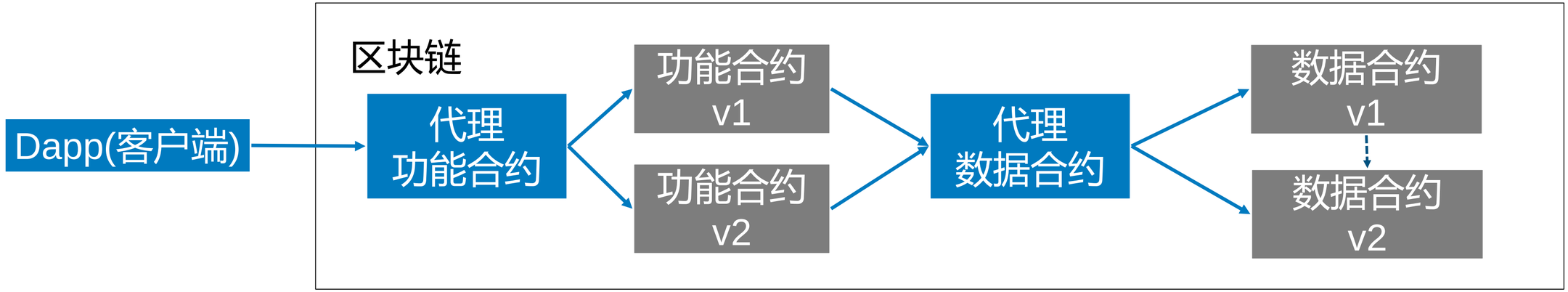
  测试方法与实践案例

功能测试

智能合约功能测试


功能测试
必要功能
业务功能
接口测试
升级功能
合约暂停
事件记录
业务流程
数据校验
参数覆盖
路径覆盖
条件组合覆盖
利用接口测试
驱动功能测试

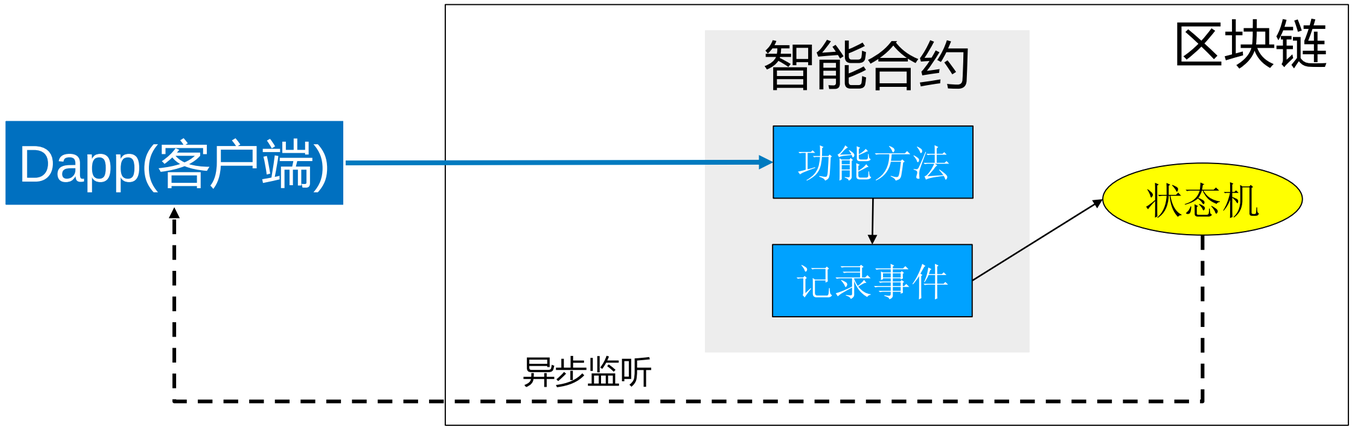
合约升级



测试维度

代理合约	功能合约	数据合约
• 地址切换到新合约 • 无法调用到旧合约	• 旧功能合约停用 • 新合约功能生效	• 数据迁移验证正确 • 旧数据合约停用

合约事件



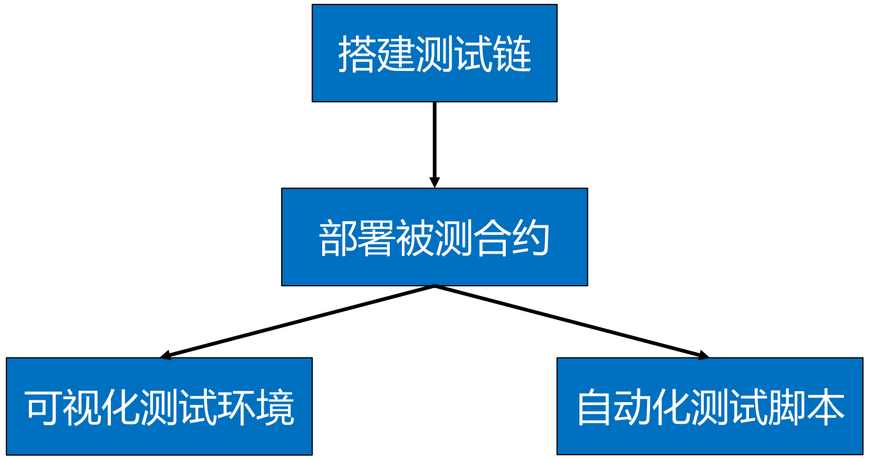
测试维度

信息完备
• 请求方 • 目标方 • 执行内容

事件校验
• 正确的调用相关方 • 完整的事件记录内容 • 准确的参数类型

智能合约功能测试

自动化测试




```python
# web3对象，区块链rpc接口
w3 = Web3(Web3.HTTPProvider("http://127.0.0.1:8545", request_kwargs={'timeout': 60}))
# # 配置钱包账户
w3.eth.defaultAccount = w3.eth.accounts[0]
# 合约地址
address = "0x80d6D92613568e3f334cCBBdFC9388C82DEF7F0A"
# 实例化合约
contract = w3.eth.contract(address=Web3.toChecksumAddress(address), abi=contractA["abi"])
# 显示所有方法
print(contract.all_functions())
# 调用合约，显示账户余额
print("当前账户余额：",contract.functions.getBalance().call())
# 调用合约 向合约转账10wei
tx_hash = contract.functions.pay().transact({"value": 10})
tx_receipt = w3.eth.waitForTransactionReceipt(tx_hash)
# 调用合约，显示账户余额
print("转账余额：",contract.functions.getBalance().call())
```

初始化

合约实例

调用合约

```
ssh://root@[redacted]/usr/bin/python3 -u /[redacted]/web3learn/quickstart.py
[<Function getBalance()>, <Function pay()>, <Function withdraw(uint256)>, <Functi
当前账户余额： 20
转账余额： 30
```

返回结果

Web3调用智能合约示例

安全验证

常见安全漏洞

类型	数量	占比
Call函数安全	41268	12.02%
条件竞争	13602	3.96%
重入攻击检测	2743	0.80%
权限控制	178925	52.13%
数值溢出	0	0.00%
事务顺序依赖	9488	2.76%
冻结账户绕过	1593	0.46%
逻辑设计缺陷	61798	18.01%
错误使用随机数	33809	9.85%

智能合约安全漏洞分布表

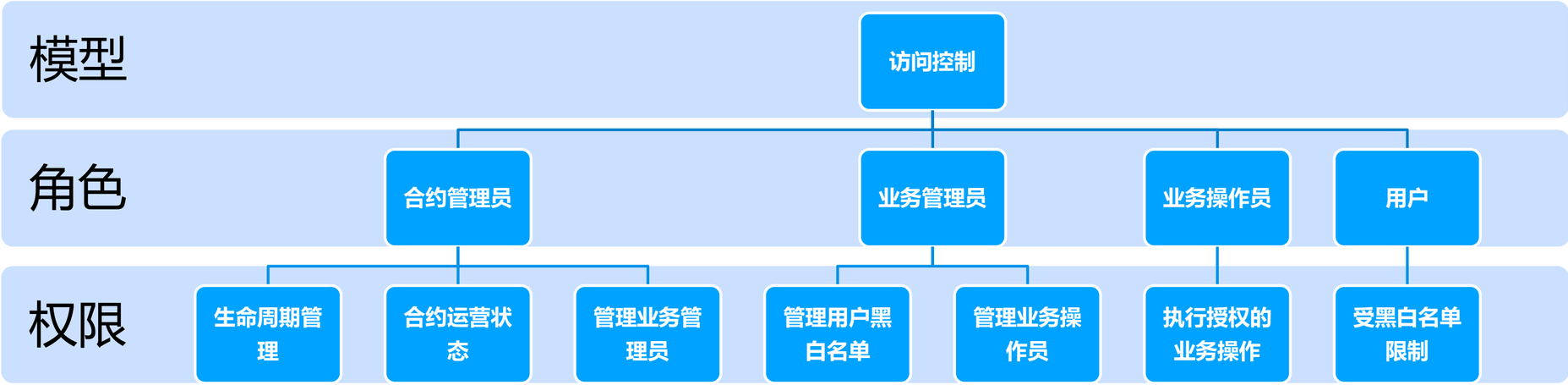
中国信通院《区块链安全白皮书(2018)》

扫描工具的局限性

	Slither	Mythril	Oyente
类型	静态	动态	静态
语言	Solidity	Solidity	Solidity
检测类型	代码规范	已知安全漏洞	已知安全漏洞
优势	可绘制关系图	贴合区块链行为	贴合区块链行为
局限性	误报率较高	一般问题识别不准	遗漏严重问题

仅支持Solidity语言　　准确率不高　　无法发现新的漏洞

RBAC权限模型与验证



测试维度

角色划分	权限设置	方法实现
• 按层级划分 • 角色划分无交叉	• 权限分工明确 • 不允许越级配置	• 所有方法均需指定其对应角色 • 先进行权限校验再执行方法

访问控制不完善

- 通过绘制合约的权限角色模型，发现合约中存在的访问控制缺陷
- 部分合约方法未判断合约是否处于暂停状态

```
function whitelist(address[] _accounts) public onlyWhitelister {
        …
    }
```

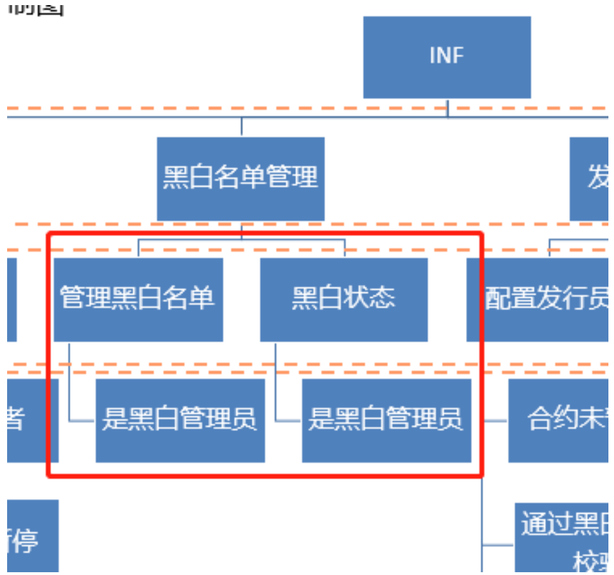
添加进白名单

```
function unWhitelist(address _account) public onlyWhitelister {
        …
    }
```

删除白名单

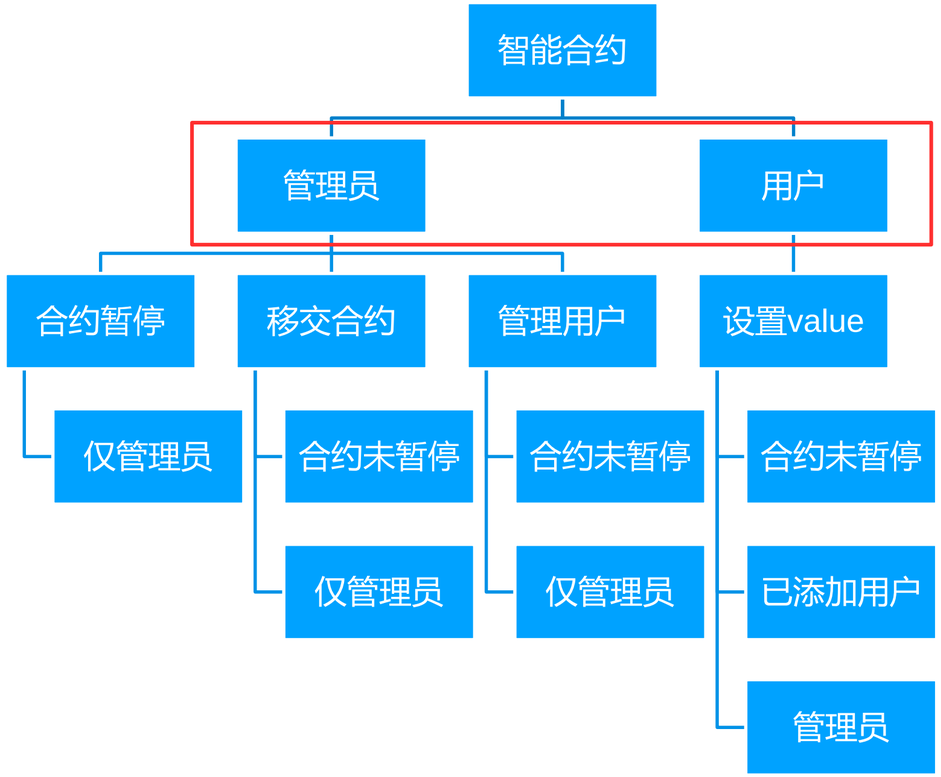
```
function whitelist(address[] _accounts) public onlyWhitelister onlyNotPause {
        …
    }

function unWhitelist(address _account) public onlyWhitelister onlyNotPause {
        …
    }
```



未合理设置角色

- 合约中仅设置管理员和用户，所有关键操作均由合约管理员执行
- 合约调用需要私钥，频繁使用管理员私钥会引发安全问题

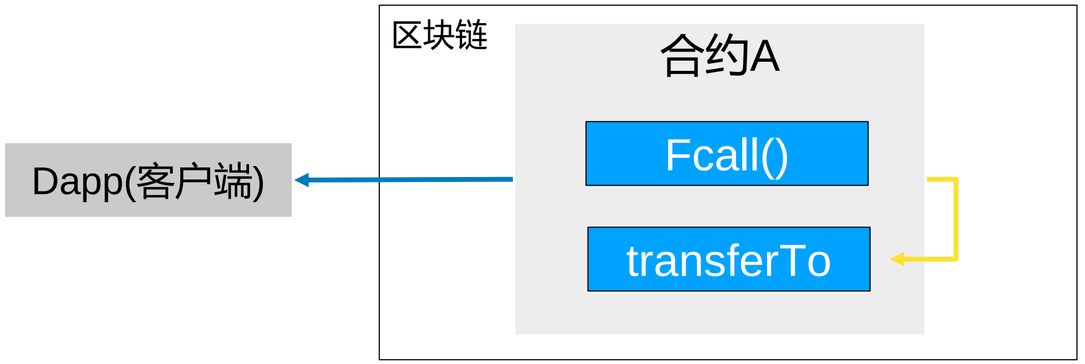


增加业务管理员和操作员角色

Call函数安全

- 自由度高，可接受任意参数
- 变更调用信息，调用发起的地址、交易金额、被调函数字符等发生改变
- 跨合约调用，调用者为当前合约，执行环境为被调用的合约运行环境

→ 注入攻击 & 权限绕过



过程	Msg.sender
用户 → A.Fcall()	用户地址
A.Fcall() → A.transferTo()	合约A的地址

```
contract A {
    address public temp1;
    uint256 public temp2;
    address owner;

    function fcall(address addr, string _callback) public {
        temp1 = msg.sender;
        temp2 = 100;
        addr.call(bytes4(keccak256(_callback)));
    }

    function transferTo(address _to, uint32 value) public {
        require(msg.sender == this.address);
        this.send(value);
    }
}
```

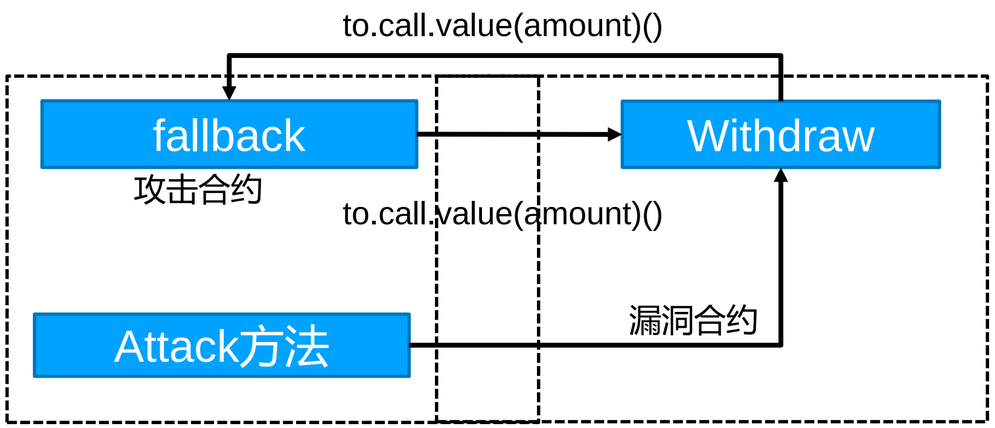
```
require(msg.sender == owner);
```

避免使用call方法
合理设置校验条件

重入攻击与防范

- 不同方法消耗的执行费用不同，有限制、无限制
- 匿名函数，方法无法匹配或转账时调用该方法
- 不规范的call方法调用
- 执行参数校验不合理

→ 重入攻击的风险



优先完成内部操作，先减后加
不允许无限制消耗gas，执行允许gas耗值

缺陷合约

```
function withdraw(address _to, uint256 amount) public {
        require(balances[msg.sender] > amount);
        require(this.balance > amount);
        _to.call.value(amount)();
        balance[msg.sender].SafeMinus(amount)
}
```

转账
扣余额

攻击合约

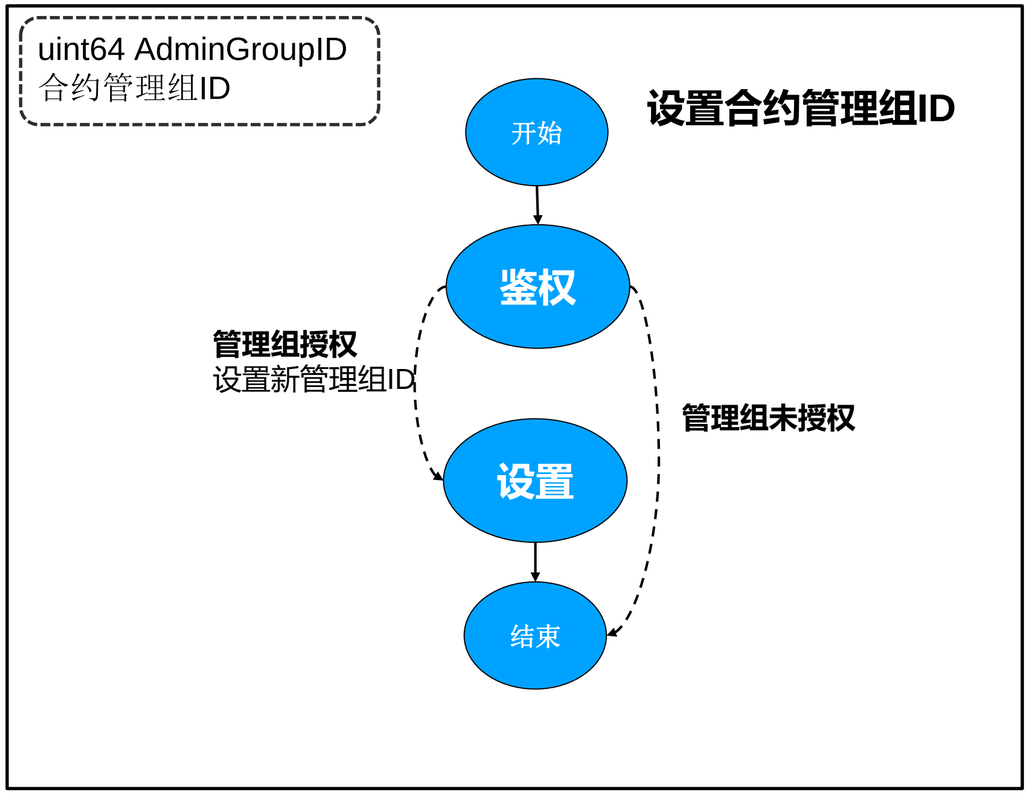
```
function () public payable {
        victim.call(bytes4(keccak256("withdraw(address, uint256)")),
        this, msg.value);
}
```

调用提款

合约流程图与人工审计

- 合约部署后，变量 AdminGroupID初始化为0（区块链共识委员会）
- 合约拥有者永远无法获得授权，合约拥有者无法配置合约
- 同时一旦合约共识委员会成员作恶，会导致合约权限泄露

- 用户应先注册自己的管理组
- 并且在合约构造函数中，将AdminGroupID初始化为已有的



合约流程图与人工审计

- 超过预期提款的问题
- 区块链不能保证合约执行的顺序性
- 攻击者会在授信变更前提走金额

- 不允许从第三方转移资产这个方法存在

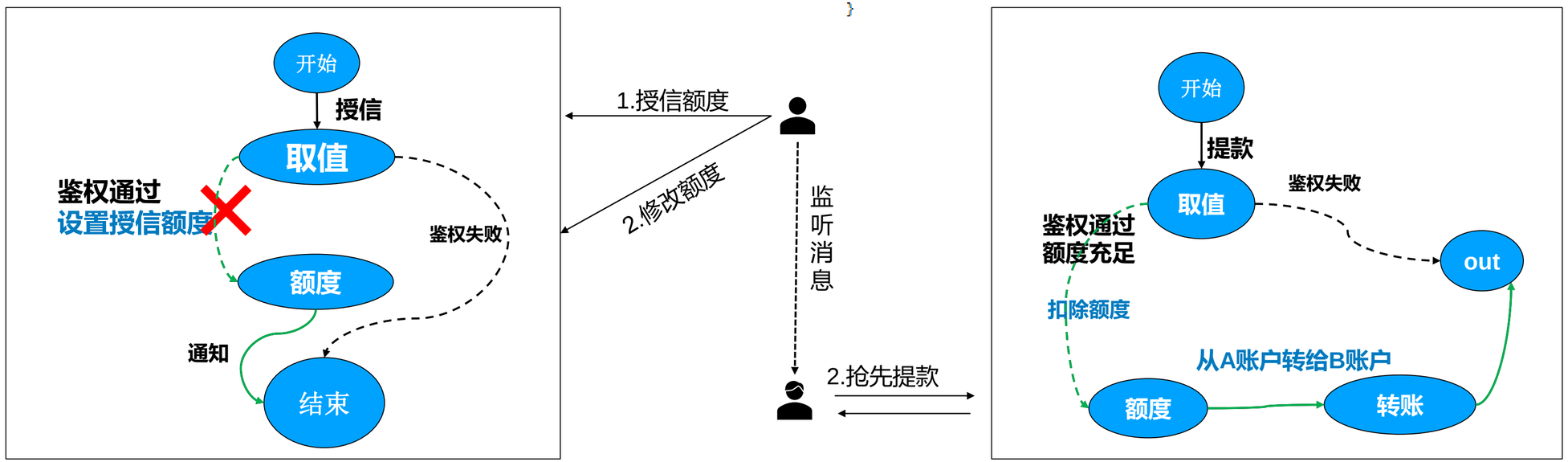
```
function approve(address _spender, uint256 _value) whenNotPaused notBlacklistec
    allowed[msg.sender][_spender] = _value;
    emit Approval(msg.sender, _spender, _value);
    return true;
}

function transferFrom(address _from, address _to, uint256 _value) whenNotPausec
    require(_to != address(0));
    require(_value <= balances[_from]);
    require(_value <= allowed[_from][msg.sender]);

    balances[_from] = balances[_from].sub(_value);
    balances[_to] = balances[_to].add(_value);
    allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
    emit Transfer(_from, _to, _value);
    return true;
}
```

给用户的授信

从授信账户提款



造成超过用户预期的提款

$

成本审计

节约成本

- 避免将合约用作数据存储
- 避免重复写入，尽可能一次写完数据
- 内存对齐，减少内存开销
- 合理定义传入参数
- 使用费用低的方法
- 减少无用代码
- 使用固定大小的字节数组
- 整合循环操作

减少存储

Name	Value	Description*
G_{zero}	0	Nothing paid for operations of the set W_{zero}.
G_{base}	2	Amount of gas to pay for operations of the set W_{base}.
$G_{verylow}$	3	Amount of gas to pay for operations of the set $W_{verylow}$.
G_{low}	5	Amount of gas to pay for operations of the set W_{low}.
G_{mid}	8	Amount of gas to pay for operations of the set W_{mid}.
G_{high}	10	Amount of gas to pay for operations of the set W_{high}.
$G_{extcode}$	700	Amount of gas to pay for an EXTCODESIZE operation.
$G_{extcodehash}$	400	Amount of gas to pay for an EXTCODEHASH operation.
$G_{balance}$	400	Amount of gas to pay for a BALANCE operation.
G_{sload}	200	Paid for a SLOAD operation.
$G_{jumpdest}$	1	Paid for a JUMPDEST operation.
G_{sset}	20000	Paid for an SSTORE operation when the storage value is set to non-zero from zero.
G_{sreset}	5000	Paid for an SSTORE operation when the storage value's zeroness remains unchanged or is set to zero.
R_{sclear}	15000	Refund given (added into refund counter) when the storage value is set to zero from non-zero.
$R_{selfdestruct}$	24000	Refund given (added into refund counter) for self-destructing an account.
$G_{selfdestruct}$	5000	Amount of gas to pay for a SELFDESTRUCT operation.
G_{create}	32000	Paid for a CREATE operation.
$G_{codedeposit}$	200	Paid per byte for a CREATE operation to succeed in placing code into state.
G_{call}	700	Paid for a CALL operation.
$G_{callvalue}$	9000	Paid for a non-zero value transfer as part of the CALL operation.
$G_{callstipend}$	2300	A stipend for the called contract subtracted from $G_{callvalue}$ for a non-zero value transfer.
$G_{newaccount}$	25000	Paid for a CALL or SELFDESTRUCT operation which creates an account.
G_{exp}	10	Partial payment for an EXP operation.
$G_{expbyte}$	50	Partial payment when multiplied by $\lceil \log_{256}(exponent) \rceil$ for the EXP operation.
G_{memory}	3	Paid for every additional word when expanding memory.
G_{txcreate}	32000	Paid by all contract-creating transactions after the *Homestead* transition.

以太坊黄页： https://ethereum.github.io/yellowpaper/paper.pdf

避免大数据存储

- 存储将消耗花费较多执行成本
- 减少不必要的存储操作

```
contract StoreTest {

    string public data;


    function storeLargeData(string _data){
        data = _data;
    }
}
```

存储数据

status	0x1 Transaction mined and execution succeed
transaction hash	0x190eed38c09577b6cd8e2daf6d…
…	
gas	3000000 gas
transaction cost	627564 gas
execution cost	547988 gas
…	
decoded input	{ "string _data": "largelargelargelargelargelargelargelargelargelargelargelargelargelargelargelargelargelargelargelargelargelargelargelargelargelargelargelargelargelargelargela…
…	

大数据

减少存储空间

status	0x1 Transaction mined and execution succeed
transaction hash	0x7502a4c7d8200ac0d2a3440be1…
…	
gas	3000000 gas
transaction cost	82297 gas
execution cost	142490 gas
…	
decoded input	{ "string _data": “small”}
…	

小数据

减少存储次数

- 存储将消耗花费较多执行成本
- 将多次存储的数据合并存储

```
contract Test {

    uint256 public amount;

    function loopstore(uint256 times){
        for(uint i = 0; i < times; ++i){
            ++amount;
        }
    }

    function storeOnce(uint256 times){
        uint256 i = 0;
        for(uint x=0; x < times; ++x){
            i++;
        }
        amount = i;
    }
}
```

循环10次存储

循环计算，1次存储

status	0x1 Transaction mined and execution succeed
transaction hash	0xe8de2be66da0b280b5d3ff0407…
…	…
gas	3000000 gas
transaction cost	46359 gas
execution cost	24895 gas
…	…
decoded input	{ "uint256 times": "10" }
…	

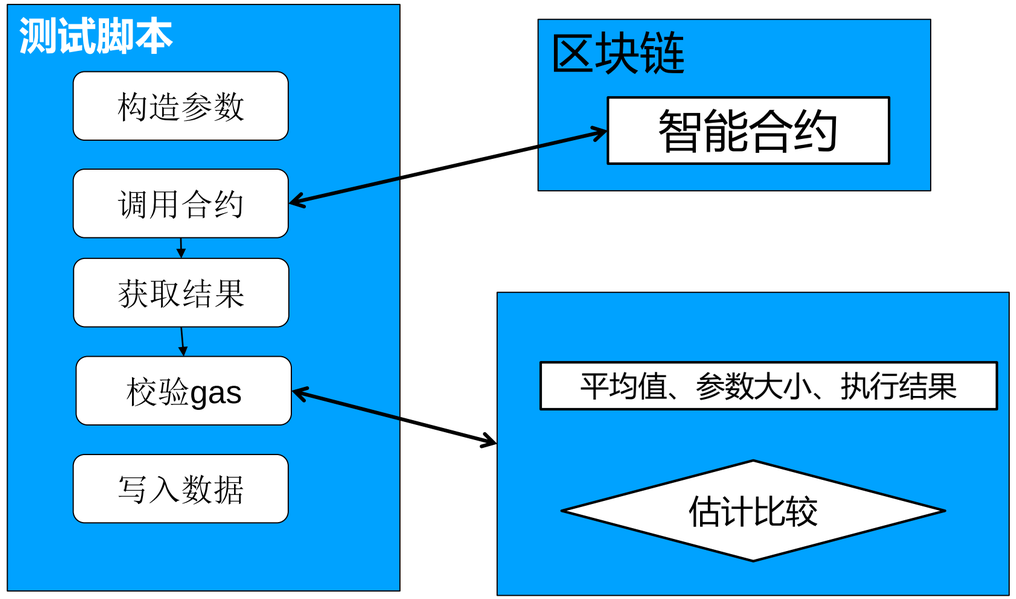
循环存储

status	0x1 Transaction mined and execution succeed
transaction hash	0xea97b8665a00a6fc9fb5229b3…
…	
gas	3000000 gas
transaction cost	22622 gas
execution cost	1158 gas
…	
decoded input	{ "uint256 times": "10" }
…	

单次存储

迭代中的成本测试

- 依据成本审计点，进行代码走读
- 在基准版本中，记录每次执行所花费的gas
- 在迭代版本测试中，校验对应方法的执行费用较原来的平均值浮动范围



Fid	applyid	bankcommente	Fresult	Fgas
1	894a14f0b4 f70208403b70d	BankComment 7187475	0	51
2	3edd0270b 670208403b70d	BankComment 869832	0	51
3	5895f0a1b4 070208403b70d	BankComment 8623605	0	51
4	5895f0a1b 070208403b70d	BankComment 8623605	0	51
5	6c086cd1b 5e70208403b70d	BankComment 3734697	0	51
6	6c086cd1b 5e70208403b70d	BankComment 3734697	0	51
7	8015061eb 38670208403b70d	BankCommen 33516161	0	51
8	8c6185c0b4 3af70208403b70d	BankCommen 1473262	0	51
9	07f93b61b4 12070208403b70d	BankCommer 03417447	1	0
10	07f93b61b4 12070208403b70d	BankCommer 03417447	0	51
11	17bd0770b a4870208403b70d	BankCommer 36549045	1	0
12	17bd0770b 4870208403b70d	BankCommen 36549045	0	51
13	315c0500b4 8a70208403b70d	BankCommer 226979	1	0

在数据库中记录调用参数、结果，gas值

智能合约测试总览

代码走读

- 方法角色图
- 梳理业务逻辑

功能测试

- 必备功能
- 业务功能
- 接口测试

安全验证

- 构建方法模型
- 常见漏洞审计
- 工具扫描

成本审计

- 调用参数
- 空间复杂度
- 构建成本基准

发布验证

谢谢

THANKS