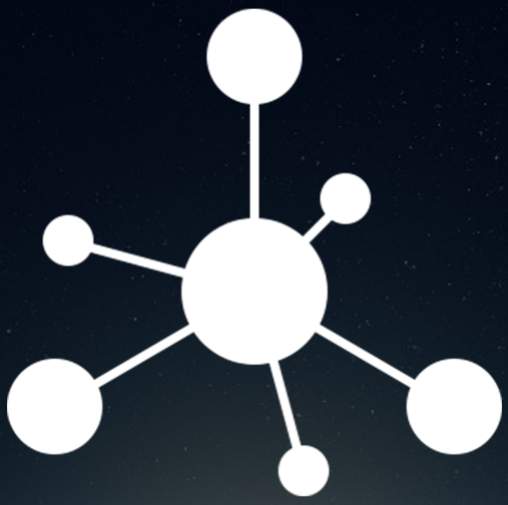


# 摩尔(MOL)

移 动 互 联 的 区 块 链 经 济

www.mol.one

# 目录

# 摩尔:移动互联的区块链经济

Alex Qian[1] Halton Xu[2] Sandy Ye[3] Harry He[4] Eddy Guo[5]
Molecule Foundation, Singapore

**摘要:** 性能瓶颈已经严重制约了以比特币和以太坊为首的区块链经济。而出块速度(Block rate)的提高和块大小(Block size)的增加以及共识效率的提升是牵引公有链向前发展的三架马车。现有的一些解决方案：分片，包括以太坊的智能合约分片，或者 Zilliqa 采用的网络分片；分层，比特币以及采用未花费支出(UTXO)为模型的山寨币采用的闪电网络；共识效率，从工作量证明(POW)到权益证明(POS)，再到拜占庭容错(BFT)，最后到 DPOS，已经到了极限。以块(block)作为数据存储单元的链已经陷入了“不可能三角”-去中心化，性能，安全。DAG(有向无环图)是一个颠覆性的解决方案，IOTA 的 M2M(machine to machine) 和 Nano 的支付都是不错的案例。摩尔 MOL 受到 Nano“区块点阵”的启发，将交易与块分离，进而摆脱“块的束缚”，处理双花采用按需 DPOS 共识。移植比较成熟的以太坊虚拟机 EVM，在 MOL 摩尔链上支持智能合约，方便 Dapp 发行代币(Token)，摩尔是 50 毫秒确认交易并且免手续费的 DAG 公有链，以此构建移动互联的区块链经济。

## 1. 区块链编年史

### 1.1 比特币和区块链 1.0

2008 年美国次贷导致的世界范围内的经济危机，让中本聪反思垄断的金融对世人的剥削与压迫，他于当年发布比特币白皮书[1]，创造了一种无政府主义的基于点对点网络的加密数字货币。人类第一次可以自己存储资产，自从比特币诞生以来，被各种传统经济学家宣判过 129 次死亡的它不断得到认可[2]，而价值也得到了上万倍的增长。

比特币设计之初考虑了安全和容量之间的权衡(Trade off)，也就是牺牲了容量以满足网络安全，这就是为什么现在的一笔交易需要 1 个小时以上的时间进行确认。

[1] Alex Qian: Blockchain developer and M.tech from IIT, Delhi, Serial Entrepreneur
[2] Halton Xu: Blcokchain architect, Full stack developer, Linux, C++, Node,
[3] Sandy Ye: Full stack and crypto wallet developer with 13 years' of coding
[4] Harry He: Full stack and blockchain engineer of MOL
[5] Eddy Guo: Blochain Engineer of MOL

### 1.2 以太坊和区块链 2.0

比特币杂志编辑 Vitalik Buterin 在 2015 年开始了另一次区块链升级的旅程，把人类认知带入到一个全新的时代。将比特币中的脚本语言抽象升级为了智能合约，基于此的 DAO(Decentralized Autonomous Organization) [3]成了未来互联网公司雏形，也因此催生了一个通证经济时代。

但是基于工作量证明机制的以太坊同样存在性能问题，一个以太坊游戏 cryptokitty 可以让整个网络瘫痪。[4]

### 1.3 DAG(有向无环图)和区块链 3.0

只有富裕国家的人民才能拥有世界上大多数的计算机，比如说 75%的计算机分布在前 20 国家一样。178 国家共分享了剩余的 25% 的世界计算机。印度直接进入了移动互联网时代，手机银行在非洲已经取代传统的银行。区块链 1.0-2.0 专注在 PC 互联网的环境下提供区块链服务。然而 90%的手机用户无法很好参与到这次数字货币革命中来。这就是摩尔链的愿景，移动互联的区块链经济。

区块链的可扩容性问题严重损害了由比特币及 Ethereum 衍生的分布式经济的扩散。区块的大小和生成速度从根本上成了区块链的瓶颈，现在主流的解决方案，有向无环图（DAG）和闪电网络以及分层。有向无环图比较适合支付，加上摩尔 MOL 对于智能合约的深度创新，使得摩尔成为免交易费的 DAG 公有链，交易最快确认速度可达 50 毫秒，只受限于带宽和网络延迟。吐吞量因为并发，理论上没有限制。

## 2. 为什么是摩尔 MOL？

### 2.1 小额支付

小额支付的特点相对应的是支付场景的日常性与平民化。而小额支付却是现有数字货币支付的最大痛点，比特币的支付需要平均 1 个小时或者更久，平均每笔 100 美金，以太坊的平均确认时间为 10 分钟，成本平均要 2 美金。逻辑上讲你是不能在付款台等待 10 分或者 1 个小时的确认时间的。如果在网上看一个视频或者在早餐店买一杯豆浆都是无法接受的。

所以，即时确认和免交易费是数字货币流通的巨大痛点，因此摩尔 MOL 应运而生，其 50 毫秒的交易确认时间和交易免费的特点，将使其成为数字货币流通的基石。

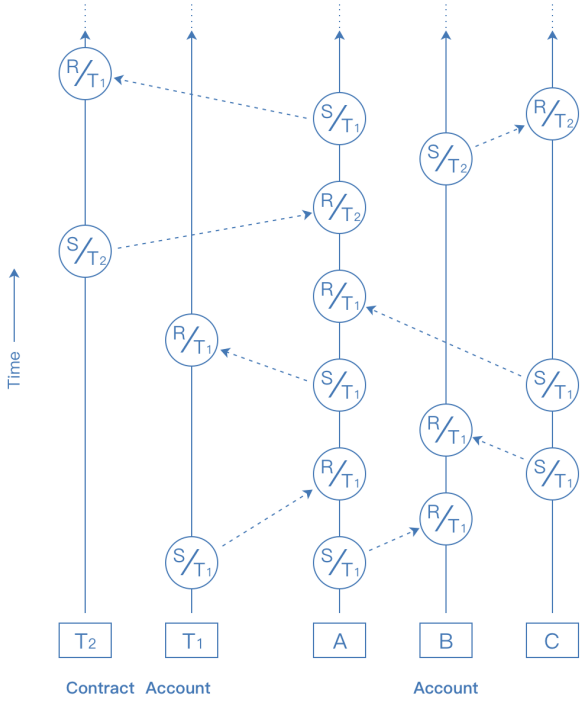
## 2.2 移动互联

由于在手机上支持小微付款，摩尔 MOL 自然地将去中心化的区块链经济推向移动互联。

游戏、内容、社交等 App 接入摩尔 MOL 就可以轻松地使用摩尔 MOL 主链即时确认与免手续费的转账、支付等服务，也可以将传统积分代币化，或者发行新的数字资产。

# 3. 摩尔 MOL 的技术特性

## 3.1 设计原则




MOL 摩尔的架构

在 MOL 摩尔链设计中，每一个用户和智能合约用户都有自己的链，用户 A 发起转账交易，私钥签名后，广播至网络中，用户 B 创造接收交易，签名后，进行广播，并且写到自己的链里。

### 3.1.1 无区块

区块与矿工本质上制约了区块链的性能，全世界在某段时间内(比特币为 10 分钟，莱特币为 2.5 分钟，以太坊 15 秒) 所有的交易数据$\sum_{t=0}^{n} Tx$要被装到一个区块（模具）里。而矿工们要在这个时间里达成共识谁来打包，比特币要 10 分钟进行 POW 共识。

挑战：

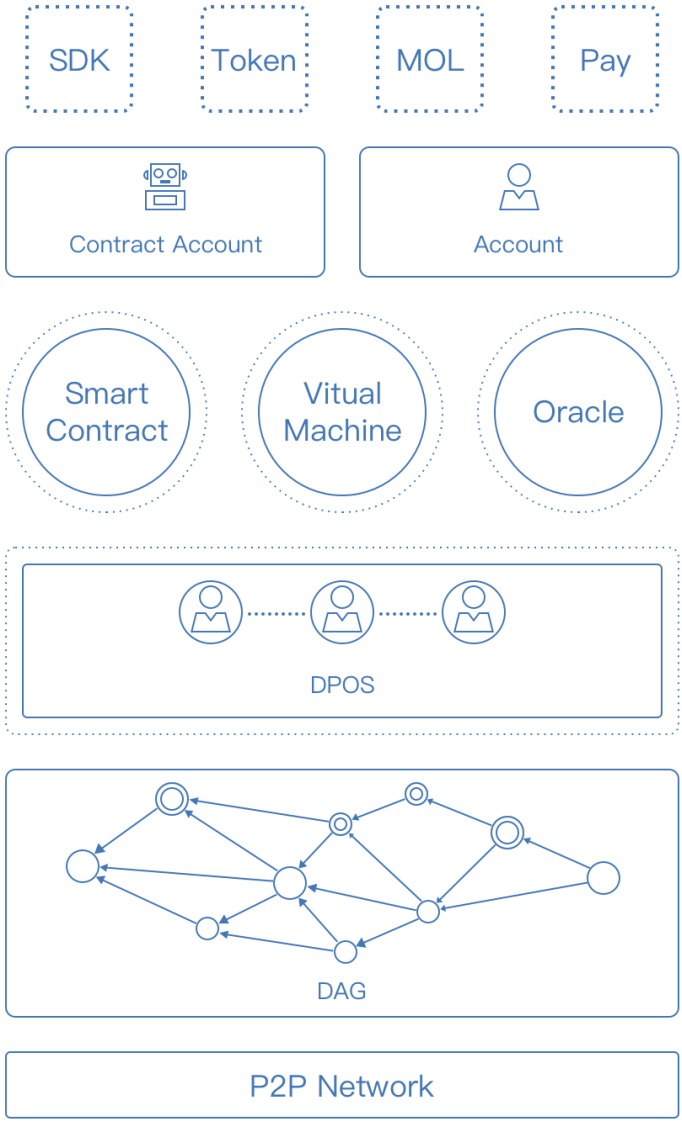
a. 对于尖锋时刻，如中国的春节或者是麦加朝圣千百万人同时发送红包，那么将用多大的一区块才能同时容纳下这些交易呢？比特币的一个交易 500 字节，一个区块大小为 1M（$1024K$ 字节）在 10 分钟内的瓶颈就是大约 2000（$\frac{1,024K}{500}$），每秒钟就是大约 3（$\frac{2000}{60*10}$）笔，而 Visa 至少可以处理 4700 笔，支付宝可以达到 10 万，后面两者都没有区块的限制。

b. 矿工被授予了不可控的权力，在区块容量超限的情况下，矿工可以对于交易具有选择权，而且由于拥堵的存在，交易费用也提高，比特币最高可达 100$/笔，而以太坊为 2$/笔。这已经远远背离 BitTorrent (比特币继承了 BitTorrent) 的免费公平的 Peer-to-Peer(对等网络)精神(人人为我，我为人人)，这将极大制约数字货币从早鸟向早期大众的大裂谷(Chasm)的穿越。

c. 共识形成打包矿工的过程的通信开销以及成本已经变得非常高昂，为了维护比特币和以太坊这两张网络，世界为此每年支出 10 亿美金左右的电力与硬件资源成本。现在全世界也只有约 500 万人左右在使用比特币，如果这个数字提升到 5000 万或者 5 亿，这样的成本是不可持续的。为了降低共识开销成本，在过去的 10 年中，从 POW 的 Asics-resilient Scrypt， X11，Cryptonight 以及一系列的权益证明算法 POS[5] 1.0， 2.0， 3.0 的 peercoin[6]， Nxt， Qutm[7]， 最后到 DPOS 的 BTS， Steem， EOS，和拜占庭容错的 BFT 俱乐部， dBFT， fBFT， pBFT 甚至是混合共识，共识方面的创新已经无以复加，而且越来越中心化。

我们还停留在中本聪 10 年前区块链的“块”的定义里，这会是刻舟求剑吗？

据此，在建设 MOL 摩尔公有链时， 我们倾向于 IOTA[8]与 Nano[9]倡导的无区块架构，让交易成为数据单元摆脱区块的限制，即无区块(lockless)。




**MOL 摩尔协议栈**

### 3.1.2 智能合约

IOTA 和 Nano 设计的缺陷在于不支持智能合约，而 MOL 摩尔从 DAG（有向无环图）的架构出发，进行创新，设计了合约账号，合约账号与正常账号有同样的功能：开户，发送，接收和改变代表，只不过合约账号是由代码控制的。

每个合约账号都是一个独立的链，从初始区块 $Block_0$， 发送与接收交易构成了无区块的区块链。

### 3.1.3 发行代币

从用户生成内容 UGC (User Generated Content) 到用户生成货币 UGC (User generated Currency)， 这个不可逆的刚需指引 MOL 摩尔团队投入研发资源致力于为去中心化的 Dapps 提供代币发行业务。

a. 摩尔将拓展比较成熟的以坊虚拟机以支持摩尔 MOL 创新的 DAG 架构 ,这样比较方便于 solidity 合约迁移至 MOL 摩尔链平台。
b. 摩尔把编译器封装成用户友好的图形界面使得任何人输入 “名字,代号,发行数量”三个参数，即可以发行属于自己的代币，降低用户的使用门槛。
c. 智能合约生成的代币，从创始区块中开始分发给普通账号，合约生成方可以根据预先设计好的兑换比例，也可以采用摩尔对换，过程类似于以太坊的发币过程。

## 3.2 共识算法: POW and DPOS

### 3.2.1 Proof of work

为了防止垃圾攻击，在每一笔交易被发送到摩尔 P2P 网络前，都需要进行一次工作量证明的运算，以提高恶意攻击的成本，整个过程会花几秒钟时间，这个时间也可以提前计算，就是用户打开钱包时即进行计算，当填写完转账数据时，POW 计算也已经完成，所以用户感知到的，是即时转账。摩尔使用的哈希算法是 ED25519&Blake2b。

### 3.2.2 DPOS

摩尔 MOL 采用 DPOS 的机制处理双花(Double-spend)。就共识效率而言，POW<POS<BFT<DPOS ,DPOS 由于有最低的共识开销成本 ,所以是效率最高的共识机制。由于双花只会发生在软件 Bug 或者是恶意攻击上，因此是按需共识（投票 0），这样可以更进一步提升工作效率，即异步共识，所以交易确认时间不依赖于共识效率。每个钱包节点可以指定及改变它们自己的代表（Representative），判断共识的方法是

$$T' = \sum_{i=0}^{n} \sum_{j=0}^{m} Token$$

$$(n = sum\ of\ Representative, m = sum\ of\ account\ under\ \ each\ representative),$$

$$T = Gensis$$

如果$T' > T$，则出现双花，

代表需要进行投票，来在两个或多个双花中选择。

## 3.3 性能

无区块的架构设计可以使交易以互联网广播的速度发送和确认，50 毫秒（1 秒=1000 毫秒）即$\frac{1}{20}$ 秒。由于交易是并发的，并且异步按需共识，理论上来讲每秒可以处理的交易量是没有限制的 unlimited。

## 3.4 跨链

2009 年起以比特币为对象的山寨链 Lite，分叉币，专注隐私的 Monero，Dash，Zcash 等，相对中心化的 Stellar 和 Ripple ，以及 2015 起基于以太坊发行的 Token 或者其竞品 Aethernity、Cardano（ADA），也有类似 IOTA，Byteball 等 DAG 链，Lisk、Ark 也自从一派，加之 NEO 与量子等生成的代币，当然又多了 EOS 系。资产在不同链上，而不同链之间的互转只有通过中心化的交易所。

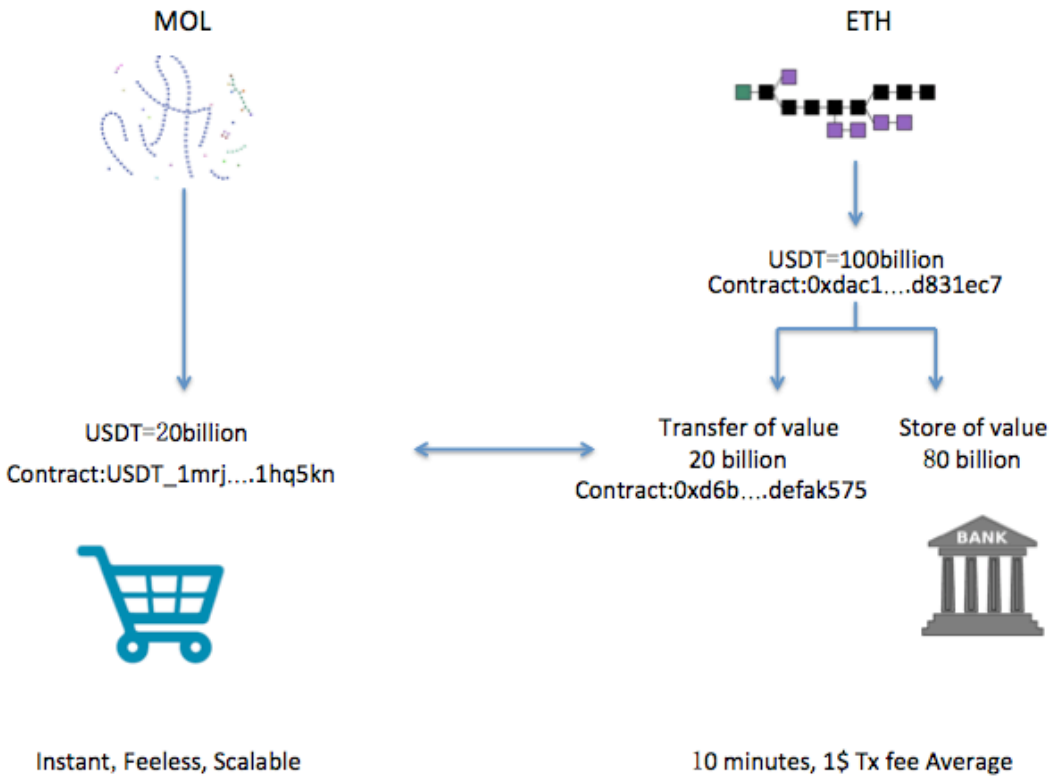
基于 MOL 主链和智能合约的跨链技术，基于下述原则：

(1) Value of Reserve 储备金

(2) 双向锚定

(3) 智能合约

(4) 价值分层



以 Ethereum(以太坊)上发行的资产 USDT 为例，假设发行 1000 亿，其中 80%以 store of value 的储值品在交易所里做交易对，这些 USDT 的交易不上链，不会产生交易手续费。但也限制了 USDT 的交易流通价值(Transfer of Value)，以太坊上发行的 token 确实缺少流通能力，以平均确认时间 10 分钟，和约 1$的转账成本，作为小额交易的流通是无法接受的。但是 Ethereum 是不能修改也没有必要修改，因为 Ethereum 最大的价值是协助企业融资，而作为类证券类存在的以太坊代币 Token 是不需要专注流通的，与比特币类似，比特币作为储值品存在，其 1 个小时以上的转账确认时间以及平均 100$的转账交易成本也是可以接受的。当一部分代币，如 USDT 以流通为目的存在时，是必须使用跨链技术，将价值转移到即时免交易费的链，例如摩尔 MOL，这样价值既进行了分层，分为 Store of value 和 Transfer of Value，以 200 亿的 USDT 转入以太坊-MOL 公证人合约锁定，完成储备金流程，在摩尔 MOL 链上建等值等额的 USDT 进行流通。

## 3.5 货币经济

### 3.5.1 staking 利息

为了保障网络安全，尽管摩尔是免交易费的，钱包节点仍然需要长期持有一定数量的摩尔，抵押的摩尔会生成利息，以每年 5.5%作为年利率。这个过程我们称之为 minting（造币）。

### 3.5.2 奖励

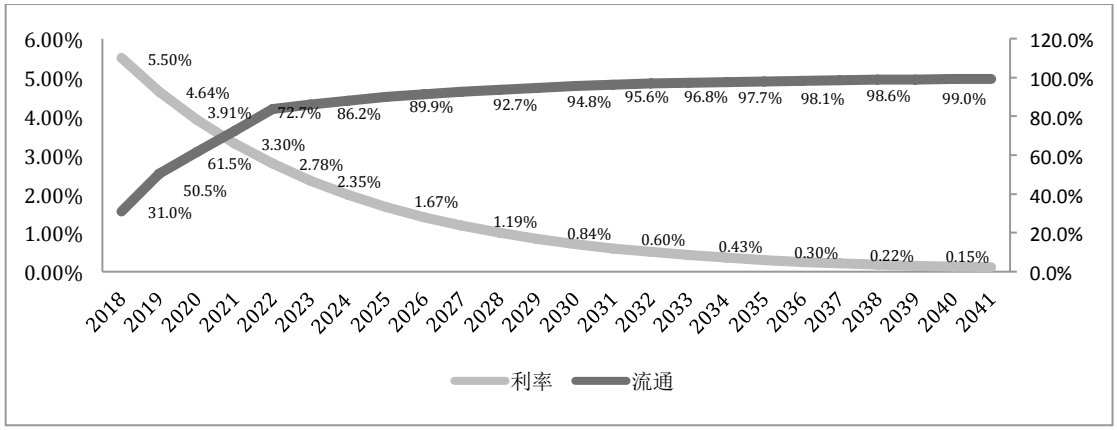
由于账号选择代表，代表会进行一个称之为“贡献证明”( proof of contribution ）的机制，为了激励代表发展更多的账号，并为账号提供更好的服务，将有一部分代币以奖励的形式授予代表。

在摩尔的全部供应中，会有一部分以锁仓利息和贡献奖励的方式以智能合约账号单独储存，定期会由代码执行根据算法自动打入钱包与代表节点。

### 3.5.3 经济模型

$$T_{n+1} = T_n * [1 + 0.8^n * 0.025]$$
$$T_0 = 0.8312$$

(注：T 为 MOL 摩尔在 n 的流通总量，初始利息为 5.5%，在 100 年时，摩尔将不再产生利息)

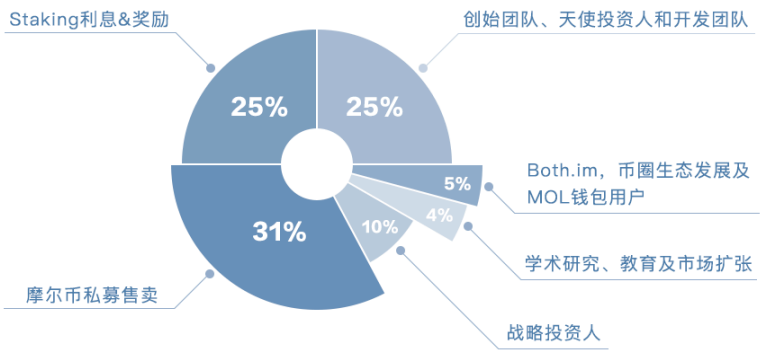



### 3.5.4 摩尔分配

总量 400 亿，摩尔链将用 4 年时间将 65%的摩尔(MOL)发到社区手中，成为真正开源的软件。

| 比例 | 分配方案 | 明细 | 说明 |
|---|---|---|---|
| 31% | 摩尔私募售卖 | 摩尔私募售卖获得的收入将会用于摩尔链基金会的运营，包括开发、市场、财务和法律咨询等。 | |
| 10% | 战略投资人 | 作为对摩尔链发展提供资源的战略投资人的回报。 | 锁仓 4 年，每年激活 1/4 |
| 25% | 创始团队、天使投资人和开发团队 | 创始团队、天使投资人以及开发团队在摩尔链的发展过程中做出了人力、物力以及技术的贡献，因此以发放摩尔币作为回报。 | 锁仓 4 年，每年激活 1/4 |
| 25% | staking 利息&奖励 | 钱包节点仍然需要长期持有一定数量的摩尔，抵押的摩尔会生成利息；为了激励代表发展更多的账号，并为账号提供更好的服务，将有一份代币以奖励的形式授予代表。 | |
| 4% | 学术研究、教育及市场扩张 | 用于支持摩尔链相关的学术研究、开发人员的教育材料、教育及市场扩张、提高对摩尔链技术的意识以及向其他开源社区进行贡献。 | |
| 5% | both.im | 用于币圈 app 的生态发展及 MOL 钱包用户。 | |

摩尔MOL分配




# 4. 总结

摩尔 MOL 采用颠覆性的有向无环图(DAG)和无区块化(Blockless)设计，使得摩尔具有 50 毫秒最快交易确认特性；其高度可扩展的并发无局限(unlimited)的性能以及免交易手续费让摩尔链天然成为小额支付、闪电支付、高频支付、博彩游戏、社群等分布式经济体代币的首选。增强型以太坊虚拟机和智能合约也让摩尔公有链成为基于免交易费的小微闪付的金融应用的魔方 ,开发者可以开发各类金融应用。

# 5. 应用场景

## 5.1 闪电支付

不像比特币的一个小时，也不像以太坊及 ERC20 代币的平均 10 分钟确认时间，摩尔 MOL 拥有 50 毫秒的闪电支付确认速度，适用于线下购物及餐饮和线上娱乐消费(看电影、打赏主播、阅读电子书、听音乐)等场景，即付即用，即付即走。

## 5.2 小额支付

对于支付宝 100 万亿的年度交易量。在 UGC(User Generated Currency)用户生成代币的时代，会有部分场景切换到数字货币，比特币的交易费平均 100 美金，而以太坊及 ERC20 的代币是 2 美金，它们不适合用在小额支付的场景，而免交易手续费的摩尔却与这个场天然匹配。

### 5.3 积分代币化

Dapp 采用现有方案解决，如 Ethereum 进行积分代币化，受到交易费和确认时间的影响导致积分代币过程中出现大量的交易上损耗和延迟 ,进而不能将代币的经济逻辑直接写入合同。

### 5.4 游戏筹码

游戏(比如麻将、斗地主等)需要免费即时的交易，现有游戏可以直接使用摩尔链将筹码以代币形式发放给用户。

## 6. References:

[1] Bitcoin: A Peer-to-Peer Electronic Cash System by Satoshi Nakamoto
[2]https://www.caseyresearch.com/they-killed-bitcoin-129-times-each-time-it-came-back-even-stronger/
[3] A next generation smart contract & decentralized application by vitalik buterin
[4] CryptoKitties craze slows down transactions on Ethereum
http://www.bbc.com/news/technology-42237162
[5] Non-Interactive Proofs of Proof-of-Work
[6] https://peercoin.net/
[7] Qtum white paper
[8]
https://www.reddit.com/r/Iota/comments/6h3sc8/what_are_the_cons_of_iota/
[9] Nano: A Feeless Distributed Cryptocurrency Network

# 7. Appendixes 附件

## 7.1 智能合约新定义数据结构

### 创建账号

```
open {
account:mol_1ob1gzhrpxdmoka7szc9iy7jtfnuysz8kkuu3ga6fni3ypquu3e7bfi3t9cr,
source:486C0A0A13F06C5CFC01684CFC1E612050827C4C8080695524A69AD0175E37A8,
representative:mol_3ytentj15q44he4c778317r868wdttwufp96fsccjux4tuqc59ojgrwn6d4w,
work:5806aa8b5b317585,
codeHash:04F952F3BE920D09E7CD7AE648E8293A4445CC1C27F2D210E1C78EAF9952EF43,
type: open,
signature:FCDB24B64DC82AD9EFE8DAA333054E58BC4D4A2A8F61E739DB7A82901976586D1BED
ECE5B206734C770CC6FAD13A79836780AEFDBAA4D92606929D6656F03603
}
```

### 创建合约

```
contract {
account: mol_3mfo37obgskt7gcy4n63efie1m8r45wij5fb734se1g1soc85fza5ffpzrpm,
source:CDB5096A97665A2B95E150816360C04CD810F9088DA928459601C0CD5461B7E8,
representative:mol_3o4p44jtc6ij9i789bzzze14xdw6sx9tziejiki54u1dmrq71fkmqc7fr      o68,
work: f04f53f1d80a213b,
init: PUSH1 0 CALLDATALOAD SLOAD NOT PUSH1 9 JUMPI STOP JUMPDEST PUSH1 32
CALLDATALOAD PUSH1 0 CALLDATALOAD SSTORE
data: 2020202020
type: contract,
signature:F153D770DD834BAAEEC57BD2262C7D78CF771C899B9AC3F230C6DDFE8CE1F02C472F
2F9345A4845D6D5DA9EDED3A038149295208ED95C9CDFF894C53DD8BFA01
}
```

### 发送合约

```
csend {
account:mol_3xh5yoa4imaat66mmg6y7fr8baks45bnwyfytapygo8nda73aqpio3yidcgm,
source:1F022C454D2BCAD35BB4F3FF8C5B9B263B0BF016563FADF569E395DE8754B5C
E,
representative:mol_1nfdb7scmjta76qoafn1zc8ri6o9hd9k44azg8obgkbgnogxr5bt3gfyx8mk,
work: 4c71b5959ff864d5,
type: csend,
signature:05D60DFCBFA55EBFBF644CB6E801B21C2C10558D815DA5859B4C46D4C2663
264DD1FB8E52C0290FD958B099F806BB28407EA37CE40D7517B0C5A276EB2C3D107
}
```

### 接收合约：

```
creceive {
account:mol_1oi8ax4kjb7hrn8un5bo3f349dsp1yme4q5ksihokdwx314gw96t856zrery,
source:BDE1AA3A6118A37E74DECF32906C372E482B5678BB3BBB635238AB42CEAE1C
FE,
representative:mol_1murykq3jduwbnojjczsgx5brqf7awnsufk6rtyxsfnbpxfzrofi66i9ezih,
work: 843f6a1593a5c1b2,
type: creceive,
signature:5CF02E89078DEAA4C34960D233B49C0468BA157544DF1AA491BDEE43EDC5B
340762DD4868BB9EE3E35A99408F7C39809DFC9D84F2B5167FBACE29A6FCD6CE70B
}
```

## 7.2 摩尔智能合约（发行代币）新定义函数类型

Name（名称）

function name() view returns (string name)

Symbol (符号)

function symbol() view returns (string symbol)

Decimal(小数点)

function decimals() view returns (uint8 decimals)

Total supply (总额)

function totalSupply() view returns (uint256 totalSupply)

Balanceof (余额)

function balanceOf(address_owner) view returns (uint256 balance)

Transfer(转账)

function transfer(address_to, uint256_value) returns (bool success)

Transferfrom(转出)

function transferFrom(address_from, address_to, uint256_value) returns (bool success)

Allowance（批准额度）

function allowance(address_owner, address_spender) view returns (uint256 remaining)

Event 事件

Transfer (转账)

eventTransfer(address indexed _from, address indexed _to, uint256 _value)

Approval(批复)

eventApproval(address indexed _owner, address indexed _spender, uint256 _value)

## 7.3 摩尔 MVM 对以太坊 EVM 的改变（指令集）

0s: Stop and Arithmetic Operations（停止和代数运算指令集）

| 指令代码 | 助记词 | EVM 语义 | MVM 语义 |
|---|---|---|---|
| 0x00 | STOP | 停止 | 相同 |
| 0x01 | ADD | 相加 | 相同 |
| 0x02 | MUL | 相乘 | 相同 |
| 0x03 | SUB | 减 | 相同 |
| 0x04 | DIV | 除 | 相同 |
| 0x05 | SDIV | 整除 | 相同 |
| 0x06 | MOD | 模(余) | 相同 |
| 0x07 | SMOD | 代符号求模 | 相同 |
| 0x08 | ADDMOD | 相加后求模 | 相同 |
| 0x09 | MULMOD | 相乘后求横 | 相同 |
| 0x0a | EXP | 幂去处 | 相同 |
| 0x0b | SIGNEXTEND | 符号扩展 | 相同 |

10s: Comparison & Bitwise Logic Operations(比较和位运算指令集)

| 指令代码 | 助记词 | EVM 语义 | MVM 语义 |
|---|---|---|---|
| 0x10 | LT | 小于比较 | 相同 |
| 0x11 | GT | 大于比较 | 相同 |
| 0x12 | LGT | 带符的大于比较 | 相同 |
| 0x12 | SGT | 带符的小于比较 | 相同 |
| 0x14 | EQ | 相等比较 | 相同 |
| 0x15 | ISZERO | 是否为零 | 相同 |
| 0x16 | AND | 并 | 相同 |
| 0x17 | OR | 与 | 相同 |
| 0x18 | XOR | 与或 | 相同 |
| 0x19 | NOT | 非 | 相同 |
| 0x1a | BYTE | 取字节 | 相同 |

20s: SHA3 (SHA3 指令集)

| 指令代码 | 助记词 | EVM 语义 | MVM 语义 |
|---|---|---|---|
| 0x20 | SHA3 | 计算 SHA3 | 相同 |

30s: Environmental Information(环境信息指令集)

| 指令代码 | 助记词 | EVM 语义 | MVM 语义 |
|---|---|---|---|
| 0x30 | ADDRESS | 获取账户地址 | 相同 |
| 0x31 | BALANCE | 获取余额 | 相同 |
| 0x32 | ORIGIN | 获取发送者地址 | 不同 |
| 0x33 | CALLER | 获取调用者地址 | 相同 |
| 0x34 | CALLVALUE | 获取转账金额 | 相同 |
| 0x35 | CALLDATATOTAL | 获取参数数据 | 相同 |
| 0x36 | CALLDATASIZE | 获取数据大小 | 相同 |
| 0x37 | CALLDATACOPY | 参数拷贝到内存 | 相同 |
| 0x38 | CODESIZE | 获取代码大小 | 相同 |
| 0x39 | CODECOPY | 代码拷贝到内存 | 相同 |
| 0x3a | GASPRICE | 获取燃料价格 | 不同 |
| 0x3b | EXTCODESIZE | 获取代码大小 | 相同 |
| 0x3c | EXTCODECOPY | 代码拷贝到内存 | 相同 |
| 0x3d | RETURNDATASIZE | 数据大小 | 相同 |
| 0x3e | RETURNDATACOPY | 数据拷贝到内存 | 相同 |

40s: Block Information(区块信息指令集)

| 指令代码 | 助记词 | EVM 语义 | MVM 语义 |
|---|---|---|---|
| 0x40 | BLOCKHASH | 获取区块的哈希 | 不同（不需要） |
| 0x41 | COINBASE | 受益人地址 | 不同（不需要） |
| 0x42 | TIMESTAMP | 时间戳 | 不同（不需要） |
| 0x43 | NUMBER | 区块编号 | 不同（不需要） |
| 0x44 | DIFFICULTY | 区块的难度 | 不同（不需要） |
| 0x45 | GASLIMIT | 燃料限额 | 不同（不需要） |

50s: Stack, Memory, Storage and Flow Operations(栈、内存、存储、控制流操作指令集)

| 指令代码 | 助记词 | EVM 语义 | MVM 语义 |
|---|---|---|---|
| 0x50 | POP | 弹出数据 | 相同 |
| 0x51 | MLOAD | 内存加载 word(M) | 相同 |
| 0x52 | MSTORE | 保存 word 到内存 | 相同 |
| 0x53 | MSTORE8 | 保存字节到内存 | 相同 |
| 0x54 | SLOAD | 加载一个 word(S) | 相同 |
| 0x55 | SSTORE | 保存 word 到存储 | 相同 |
| 0x56 | JUMP | 跳转指令 | 相同 |
| 0x57 | JUMPI | 条件跳转指令 | 相同 |
| 0x58 | PC | 计数器的值 | 相同 |
| 0x59 | MSIZE | 获取内存大小 | 相同 |
| 0x5a | GAS | 获取可用燃料数 | 不同 |
| 0x5b | JUMPDEST | 跳转目的地 | 相同 |

60s & 70s: Push Operations(压栈操作指令集)

| 指令代码 | 助记词 | EVM 语义 | MVM 语义 |
|---|---|---|---|
| 0x60 | PUSH1 | 1 字节压入栈顶 | 相同 |
| … | … | … | … |
| 0x7f | PUSH32 | 32 字节压入栈顶 | 相同 |

80s: Duplication Operations(复制操作指令集)

| 指令代码 | 助记词 | EVM 语义 | MVM 语义 |
|---|---|---|---|
| 0x80 | DUP1 | 1 对象复制 | 相同 |
| … | … | … | … |
| 0x8f | DUP16 | 32 对象复制 | 相同 |

90s: Exchange Operations(交换操作指令集)

| 指令代码 | 助记词 | EVM 语义 | MVM 语义 |
|---|---|---|---|
| 0x90 | SWAP1 | 交换 1 和 2 对象 | 相同 |
| … | … | … | … |
| 0x9f | SWAP16 | 交换 1 和 17 对象 | 相同 |

a0s: Logging Operations(日志操作指令集)

| 指令代码 | 助记词 | EVM 语义 | EVM 语义 |
|---|---|---|---|
| 0xa0 | LOG0 | 不设主题 | 相同 |
| … | … | … | … |
| 0xa4 | LOG4 | 4 个主题 | 相同 |

f0s: System operations(系统操作指令集)

| 指令代码 | 助记词 | EVM 语义 | EVM 语义 |
|---|---|---|---|
| 0xf0 | CREATE | 创建一个合约 | 相同 |
| 0xf1 | CALL | 调用另一个合约 | 相同 |
| 0xf2 | CALLCODE | 调用合约的代码 | 相同 |
| 0xf2 | RETURN | 停止执行并返回 | 相同 |

| | | | |
|---|---|---|---|
| 0xf4 | DELEGATECALL | 调用 code 保留 Tx | 相同 |
| 0xfe | INVALID | 无效指令 | 相同 |
| 0xff | SELFDESTRUCT | 删除合约 | 相同 |