EVM 对象格式（EFO）全面升级解析

作者：Dragan Rakita，Paradigm 翻译：善欧巴，本站

### EOF（EVM 对象格式）

EOF（EVM Object Format）是一组旨在改进EVM的小型EIP。它引入了一种新的字节码格式，为EVM的未来做好准备。

## EOF的优势

EOF的价值难以解释，因为它不是单一的东西，并且由于多次在分叉中被推迟以及多年的开发和研究、不同版本的演变，使得解释变得更加复杂。

本文的目的是总结这些优势并用一句话解释它们：

### 优势：

EOF允许操作码的Gas定价变化

如果燃气定价变化，传统字节码可能会表现不同。

移除燃气可观测性

这意味着移除GAS操作码，以及CALL/DELEGATECALL/STATICCALL中的燃气限制。

允许L2根据其用例改变燃气

例如，zk L2中哈希操作的高成本。

EIP-7667：提高哈希函数的燃气成本。

减少字节码大小，降低燃气使用

早期数据表明代码/初始化代码大小和燃气使用量的减少：

Uniswap-v3部署减少6.5%的初始化代码和部署代码。

部署UniswapV3Factory使用约14%更少的燃气，调用runTest使用约9%更少的燃气。

ENS DNSRegistrar部署减少约6%的初始化代码和约1.5%的部署代码。

ENS调用proveAndClaim：使用约10%更少的燃气。

允许字节码转换和可升级性

移除代码可观测性意味着移除PC, CREATE/CREATE2, EXTCODEHASH, EXTCODESIZE, EXTCODECOPY, CODESIZE和CODECOPY操作码。

如果代码更改，传统合约将无法运行。

这将允许我们在未来引入verkle时对EOF字节码进行任何形式的修改。

EOF启用操作码立即数

打开SWAPN, DUPN和EXCHANGE操作码的可能性。

这为Solidity在栈大小上提供了更多自由，解决了Solidity中的栈深度过深问题。

移除昂贵的跳转目标分析

在Reth中，分析结果与字节码一起保存，但其他客户端不一样。移除在合约执行前的跳转目标分析提高了速度。

随着分析的移除，未来我们可以增加最大字节码大小。

静态分析变得更容易

子程序强制更结构化的控制流，这使模糊测试更有效，并且可以实现线性时间的静态分析。

数据和代码分离，更容易推理。

EOF字节码可以编译成更快的字节码

EOF字节码可以编译成机器码。

未来证明EVM

字节码的版本和结构允许其可扩展性。这对L2和标准化尤其有用。

一个例子是EIP-7701：带有EOF的本地账户抽象，增加了新的头部部分。

地址空间扩展

新的EXT\*CALL操作码通过要求地址字段填充零，为未来的地址扩展做好了准备。当以太坊决定扩展地址空间时，EOF已经为这些变化做好了准备。

## 与当前EVM的集成

对于开发人员来说，一个重要的问题是实施变化所需的努力、测试成本和维护这些操作码的成本。

新的操作码不会与传统操作码冲突，EOF的验证不会触及已弃用的操作码。

EOF的编码和解码可以进行模糊测试。验证是一个新事物，在Revm中大约有500行代码，但有很多边缘情况需要覆盖，并需要集中精力在各个实现中做到正确。

创建交易用作EOF字节码的载体，类似于EOFCREATE但在执行前需要验证。

大多数操作码非常简单：

EXTCALL (0xf8), EXTDELEGATECALL (0xf9), EXTSTATICCALL (0xfb)

具有与已弃用的CALL相同的蓝图，但移除了gas\_limit和内存输出字段。

在RETURNDATALOAD引入之前（在一个很早的分叉中），CALL操作码的内存输出必须在执行CALL操作码之前设置。这不允许动态输出。

EOFCREATE和RETURNCONTRACT

是EOF的新内容，需要特殊处理。

EXCHANGE (0xe8), SWAPN (0xe7), DUPN (0xe6), DATACOPY (0xd3), DATASIZE (0xd2), DATALOADN (0xd1), DATALOAD (0xd0), RJUMP (0xe0), RJUMPI (0xe1), RJUMPV (0xe2), RETURNDATALOAD

逻辑简单，大多数只需10-20行代码实现。没有很多需要覆盖的边缘情况。

CALLF (0xe3), RETF (0xe4), 和 JUMPF (0xe5)

需要子程序栈和栈验证，复杂度大约需要20-30行代码。

需要一个开发人员约2-3个月的时间。测试工作已经开始。目前大约有2000个手写的验证测试，状态测试工作也在进行中。

变化集中在EVM中，因此与客户端其余部分的集成取决于客户端的架构以及保存字节码的位置。

EXTCODESIZE和EXTCODEHASH需要知道账户是否为EOF，并返回预定义的值（0xEF00的大小和哈希），这可能会略微改变客户端的集成方式。一个想法是将is\_eof标志保存在普通账户表中，以在调用任何EXTCODE类型的操作码时跳过字节码的加载。

## 对L2的影响

最大的问题是为什么L2不实施这些变化？我们是否应该在以太坊L1上停止EVM改进？

现实情况是，L2还没有准备好，不仅如此，它们也没有一个平台来帮助整合这些创新。字节码版本控制有助于构建L2可以使用的平台，代码可观测性的移除有助于缓解可升级性问题，燃气变化帮助ZK L2消除燃气炸弹的DDoS向量（例如，zk中的哈希值非常昂贵）。

更重要的是，EOF不仅仅是一种格式，它还需要语言（Solidity/Vyper/Huff）的支持，并需要工具链的支持才能使用。它需要一个生态系统来使用它，这种格式为L2提供了更多的稳定性，以便在其上进行创新。

## 缺点：传统字节码仍然存在

这是一个常见问题。传统字节码将永远存在，如果我们不提供替代方案，我们将陷入困境。有了替代格式的字节码，未来我们可以在状态过期发生时过渡并移除传统字节码。

## 总结

EOF不是下一个耀眼的东西，它是修复初始EVM版本遗留问题的维护EIP，除了破坏它，没有其他方法可以解决这些问题。它对于EVM的进一步发展和未来证明是必要的。