## GAS优化的一些整理

特别注意：Gas优化手段并非所有编译器版本都一致，开发时需要做前后比较。

本文档从合约实现、合约部署、合约调用、合约运行四个主要角度进行整理，覆盖从开发到运行的整个周期。后续也将依据该文档进行详细补充。

## 合约实现

变量存储

打包：使用存储槽打包变量

不打包：确定不打包的类型使用满字节，避免转换

不读：合约不读的变量不进行存储，必要可使用事件

不写：从不修改的变量定义成constant和immutable变量

定长：能固定长度的数组尽量固定长度，避免多个槽位存储长度

验证：尽量链上验证，链下存储

归0：相对0到n存储，可考虑从n到0，可回收gas

变量读取：

简单类型读取：拷到内存，尽量一次性读取，然后一次写入

复杂类型读取：酌情使用存储指针读取，避免加载不需要的数据进内存

私有化：不必要公开的变量声明私有

数据类型：

Bool：存储大量布尔值使用位图

Uint：时间戳使用uint48已足够

String：尽量小于31个字节

数组：

可使用映射替代,避免冗余长度检查

必须使用数组时考虑遍历次数限制，可迭代数组

索引保证下可使用unsafeAccess，避免长度检查

函数：

只被一个函数调用的函数可考虑合并，减少调用跳转

构造函数和管理员函数考虑使用payable，减少msg.value的判断

函数return参数命名，使用命名赋值返回

函数选择器优化：使用前导0更多选择器

函数修改器优化，内容可考虑封装成函数（修改器部署成本高于函数，调用成本低于函数）

参数使用成本更低的calldata代替memory

避免calldata中使用有符号整数

链上读取计算，链下消费

错误：使用自定义错误，无其他参数是仅返回4字节选择器，成本更低

## 合约部署

元数据：

不需要元数据：

可使用--no-cbor-metadata设置部署时不带元数据，降低字节码存储成本

需要元数据：

考虑通过注释寻找0值更多的IPFS地址，使存储更低

代理：

使用代理模式降低大批量部署成本：最小代理、元数据代理、UUPS、透明代理

自毁：

合约使用一次部署请自毁

## 合约调用

预热：使用ERC2930访问事务列表预访问

整合：

合并合约：单合约不拆分，避免跳转

合并交易：

Multicall

元交易

抽象账户

缓存：外部调用数据酌情缓存

替换：

库替换：选择使用gas更少的依赖库，openzeppelin、solmate、solady

签名替换：使用签名替换merkle和空投

转移：

使用Layer2等扩展技术

代币：

ERC777转账钩子触发，避免多次交易

ERC2612签名批准+转账

收款支持fallback received，避免定义函数

考虑使用成本更低的ERC1155、ERC6909

## 合约运行

循环判断：

For循环限制次数+循环内部gas优化

do-while优于for循环

If-else尽量不取反，使用严格等式，避免隐式操作码

运算：

加法：自增使用++i代替i++

减法+加法+乘法：必要时不检测溢出

乘法+除法：使用位运算替代

等于：超过32字节使用哈希比较

逻辑||与逻辑&&：

断路特性，便宜的表达式优先

幂运算&对数运算：

提前定义常量

## 设计

合理使用模式：提款模式、转嫁成本风险

优化器：runs大时运行gas低，低时部署gas低

考虑使用Yul开发