可以从智能合约入手，优化代码，使其大大的降低gas的费用，那么如何降低gas的费用呢，以下我总结了几点

1. 删除不必要的库Solidity库

我们引入的库通常只需要用到其中的部分功能，这意味着其中可能会包含大量对于你的智能合约而言其实是冗余的solidity代码。如果可以在你自己的合约里安全有效地实现所依赖的库功能，那么就能够达到优化solidity合约的gas利用的目的

2. 如何选择变量数据类型存储

不同数据类型的存储消耗不同。在满足业务的情况下，我们应该选择 gas 消耗更小的数据类型。

在没有办法将多个变量放入同一个插槽时，尽量使用 256 位的变量，例如 uint256 和 bytes32

在使用小于 32 字节的变量数据类型时，合约的 gas 使用量可能会高于使用 32 字节的类型。这是因为 EVM 每次操作 32 个字节， 所以如果变量大小比 32 字节小，EVM 必须执行额外的操作以便将 32 字节大小缩减到到所需的大小

我们将如下两个合约部署在 测试链上：

//消耗69324 gas

contract A{

uint256 age=0;

}

//消耗69825 gas

contract B{

uint32 age=0;

}

可以明显看到uint32部署费用比uint256的还要高

此外，在 EVM 执行计算也需要额外的操作，除 uint256 之外的其他 uint 类型在计算时需要耗费额外的 gas 进行转换

3.使用简短的原因字符串

将错误原因字符串与require语句一起附加，以便更容易理解contract调用失败的原因。但是，这些字符串在部署的字节码中占用空间。每个原因字符串至少需要32个字节，因此请确保您的字符串符合32个字节，否则会变得更加昂贵

4.事件

没有参数的事件是750 GAS。理论上每个附加参数将增加256个GAS，但事实上，它会更多

5.哈希

你可以使用智能合约中的几个内置哈希函数：keccak256，sha256和ripemd160。参数越多，消耗的气体越多。耗气量：ripemd160> sha256> keccak256。因此，如果没有其他目的，建议使用keccak256函数

6.显式声明Solidity合约函数的可见性

显式声明函数的可见性不仅可以提高智能合约的安全性，同时也有利于优化合约执行的gas成本。例如，通过显式地标记函数为外部函数（External），可以强制将函数参数的存储位置设置为calldata，这会节约每次函数执行时所需的以太坊gas成本。

7.编译合约时使用优化器

使用编译器 solc 编译合约时启动优化器 optimizer，它将简化复杂的表达方式，能减小编译后的合约字节码大小，从而减少部署时的 gas 消耗，同时也能减少合约调用时的消耗

solc --optimize --optimize-runs 200

运行次数 --optimize-runs 指定了部署的代码的每个操作码在合同的生命周期内被执行的大致频率。这意味着它是代码大小（部署成本）和代码执行成本（部署后的成本）之间的一个折衷参数。次数越小，编译出的字节码越小，但是调用该合约函数可能需要更多 gas。

此外还有基于 Yul 的优化器，更加强大，因为它可以跨函数调用工作

8.内联汇编打包变量

编写内联汇编 (Inline Assembly) ，手动将多个变量堆叠在一起，打包到单个插槽中。

语法：使用 assembly{ ... } 来嵌入汇编代码段

// 编码时将多个变量一起储存。

function encode(uint64 \_a, uint64 \_b, uint64 \_c, uint64 \_d) internal pure returns (bytes32 x) {

assembly {

let y := 0

mstore(0x20, \_d)

mstore(0x18, \_c)

mstore(0x10, \_b)

mstore(0x8, \_a)

x := mload(0x20)

}

}

function decode(bytes32 x) internal pure returns (uint64 a, uint64 b, uint64 c, uint64 d) {

assembly {

d := x

mstore(0x18, x)

a := mload(0)

mstore(0x10, x)

b := mload(0)

mstore(0x8, x)

c := mload(0)

}

}

这种方式虽然节省了 gas，但是牺牲了代码的可读性，容易出错

9.无需使用默认值初始化变量

// 部署消耗 gas 67054

contract Test {

uint256 x;

}

// 部署消耗 gas 67912

contract Test {

uint256 x = 0;

}

10.操作合约和数据合约分离

在使用工厂合约创建合约的情况下，可以将创建的合约分离为操作合约和数据合约。

操作合约只创建一次，工厂合约每次只创建数据合约，而不是每次都创建一整个合约，从而减少 gas 消耗。