Gas Usage

Yao Sun
yao@nuco.io

19 September, 2017

Estimated gas usage is concerned with the gas costs for immediate operation of the sale, and estimated future usages (Savings, Token), we provide the costs and some context as to the reasoning behind the costs.
Results were derived from the following setup:

- TestRPC v4.1.1 (ganache-core: 1.1.2)
- Truffle v3.4.9
- web3 v0.4.12
- NodeJS v8.5.0
- NPM v5.3.0

1 Definitions

We first define the units used in this report

- e - Ether
- ae - atto-Ether, also known as wei(w), $1w = 1 * 10^{-18}e$
- ne - nano-Ether, also known as gwei/Giga-wei(gw), $1gw = 1 * 10^{-9}e$
- v - The price in terms of ether value to CAD. (Around \$370)

The default costs parameters were set to the following on September 19th, 2017:

- $g_{low} = 5.0gw$
- $g_{med} = 20.0gw$
- $g_{high} = 24.504381517gw$

2 Sales

Referring to Table 1, the sales contracts are relatively simple, the only point of interest being that our gas costs for ether deposits are higher than the ambient cost ($2300g$), therefore we must set a custom gasLimit. Recommended gas limit is $100000g$, because it is an easy to remember number far greater than $48100g$.

Table 1: Costs in the sales mechanism

Function	**GasUsed**	w_{low}	w_{med}	w_{high}	$\$_{low}$	$\$_{med}$	$\$_{high}$
deployment	2.37×10^6	1.19×10^{16}	4.74×10^{16}	5.81×10^{16}	4.33	1.73×10^1	2.12×10^1
init()	1.04×10^5	5.19×10^{14}	2.08×10^{15}	2.54×10^{15}	1.89×10^{-1}	7.57×10^{-1}	9.28×10^{-1}
()	4.81×10^4	2.40×10^{14}	9.61×10^{14}	1.18×10^{15}	8.77×10^{-2}	3.51×10^{-1}	4.30×10^{-1}

Table 2: Savings contract costs

Function	**GasUsed**	w_{low}	w_{med}	w_{high}	$\$_{low}$	$\$_{med}$	$\$_{high}$
deployment	1.05×10^6	5.24×10^{15}	2.10×10^{16}	2.57×10^{16}	1.91	7.65	9.37
changeOwner	4.39×10^4	2.19×10^{14}	8.78×10^{14}	1.08×10^{15}	8.01×10^{-2}	3.20×10^{-1}	3.92×10^{-1}
start	5.56×10^4	2.78×10^{14}	1.11×10^{15}	1.36×10^{15}	1.01×10^{-1}	4.06×10^{-1}	4.97×10^{-1}
deposit	7.70×10^4	3.85×10^{14}	1.54×10^{15}	1.89×10^{15}	1.40×10^{-1}	5.62×10^{-1}	6.88×10^{-1}
bulkDepositTo (50)	2.82×10^6	1.41×10^{16}	5.65×10^{16}	6.92×10^{16}	5.15	2.06×10^1	2.52×10^1
multiMint (100)	2.30×10^6	1.15×10^{16}	4.60×10^{16}	5.63×10^{16}	4.19	1.68×10^1	2.06×10^1
$withdraw_{special}$	8.28×10^4	4.14×10^{14}	1.66×10^{15}	2.03×10^{15}	1.51×10^{-1}	6.04×10^{-1}	7.40×10^{-1}
$withdraw_{monthly}$	5.32×10^4	2.66×10^{14}	1.06×10^{15}	1.30×10^{15}	9.71×10^{-2}	3.88×10^{-1}	4.76×10^{-1}
$withdraw_{36+1}$	1.99×10^6	9.96×10^{15}	3.98×10^{16}	4.88×10^{16}	3.63	1.45×10^1	1.78×10^1
sendTransaction	2.10×10^4	1.05×10^{14}	4.20×10^{14}	5.15×10^{14}	3.83×10^{-2}	1.53×10^{-1}	1.88×10^{-1}

3 Savings

Referring to 3, which is the closest public sales (but may not be accurate entirely for our pre-sale). Due to the fact that these are existing ICOs, their userbase may have expanded since the ICO, we take the lower end of the range and set $n_{users} = 5000$.
There are two ways to deposit tokens into the savings contract. The first is **bulkDepositTo**, which is used for users that have received liquid tokens, but want to deposit them. The second is **multiMint**, used by us to mint users of the sale that have deposited ether with the *intent* for savings $n_{mm} = 100$, $n_{bulk} = 50$.

ICO	**Users**
OmiseGO	360953
EOS	84246
Golem	63328
Status	38785
0x	20734
Bancor	12739
Gnosis	5667
FunFair	4936

Figure 1: Existing Post-ICO Token Holders Amount

Table 3: Token contract costs

Function	**GasUsed**	w_{low}	w_{med}	w_{high}	$\$_{low}$	$\$_{med}$	$\$_{high}$
$Deploy_t$	1.95×10^6	9.77×10^{15}	3.91×10^{16}	4.79×10^{16}	3.56	1.43×10^1	1.75×10^1
$Deploy_c$	1.24×10^6	6.22×10^{15}	2.49×10^{16}	3.05×10^{16}	2.27	9.07	1.11×10^1
$Deploy_l$	1.13×10^6	5.65×10^{15}	2.26×10^{16}	2.77×10^{16}	2.06	8.25	1.01×10^1
$transfer_{zero}$	5.89×10^4	2.95×10^{14}	1.18×10^{15}	1.44×10^{15}	1.07×10^{-1}	4.30×10^{-1}	5.27×10^{-1}
$transfer_{nonzero}$	2.89×10^4	1.45×10^{14}	5.78×10^{14}	7.08×10^{14}	5.27×10^{-2}	2.11×10^{-1}	2.58×10^{-1}
$approve_{zero}$	5.27×10^4	2.63×10^{14}	1.05×10^{15}	1.29×10^{15}	9.61×10^{-2}	3.84×10^{-1}	4.71×10^{-1}
$approve_{nonzero}$	3.04×10^4	1.52×10^{14}	6.09×10^{14}	7.46×10^{14}	5.55×10^{-2}	2.22×10^{-1}	2.72×10^{-1}
$transferFrom_{zero}$	6.48×10^4	3.24×10^{14}	1.30×10^{15}	1.59×10^{15}	1.18×10^{-1}	4.73×10^{-1}	5.79×10^{-1}
$transferFrom_{nonzero}$	2.49×10^4	1.24×10^{14}	4.98×10^{14}	6.10×10^{14}	4.54×10^{-2}	1.82×10^{-1}	2.22×10^{-1}
$burn_{zero}$	6.29×10^4	3.14×10^{14}	1.26×10^{15}	1.54×10^{15}	1.15×10^{-1}	4.59×10^{-1}	5.62×10^{-1}
$burn_{nonzero}$	3.29×10^4	1.64×10^{14}	6.58×10^{14}	8.06×10^{14}	6.00×10^{-2}	2.40×10^{-1}	2.94×10^{-1}
multiMint (100)	2.98×10^6	1.49×10^{16}	5.96×10^{16}	7.30×10^{16}	5.44	2.17×10^1	2.66×10^1

$$\begin{aligned} C_{mm} &= (n_{users}/n_{mm}) * c_{multiMint} \\ &= 209.5(low) \\ &= 840(med) \\ &= 1030(high) \end{aligned} \tag{1}$$

$$\begin{aligned} C_{mm} &= (n_{users}/n_{bulk}) * c_{multiMint} \\ &= 515(low) \\ &= 2060(med) \\ &= 2520(high) \end{aligned} \tag{2}$$

The first cost is more significant to us, as liquid token holders will probably incur their own costs for depositing towards a savings contract. The other significant cost incurred in this contract is operational, **withdraw**, as we can see, for one user a monthly schedule (plus special) costs 3.63. We can derive the cost of monthly by assuming $n_{batch} = 50$.

$$\begin{aligned} C_{withdraw} = n_{users} * c_{withdraw_{36+1}} - n_{users} * 36 * c_{sendTransaction} + \\ 36 * c_{sendTransaction} \\ = 11275.90(low) \\ = 45103.60(med) \\ = 55261.80(high) \end{aligned} \tag{3}$$

4 Token

Token contract costs are all operational (for the user), the only costs incurred on us is **deployment** and **multiMint**. Similar to the multiMint for savings, we calculate the cost $n_{batch} = 100$.

$$
\begin{aligned}
C_{multiMint} &= (n_{users}/n_{batch}) * c_{multiMint(100)} \\
&= 272(low) \\
&= 1085(medium) \\
&= 1330(high)
\end{aligned}
\tag{4}
$$

5 Conclusion

By far the largest cost for us is the operational cost of the monthly withdrawals for the savings contract. However, we can mitigate this be either withdrawing at a decrease frequency, or pushing the responsibility to migrate on the user.