Ethereum Transaction Fees

Gas

Gas is the equivalent of transaction fees in Ethereum

- Each opcode of the EVM has a certain gas cost
- Limits computational cost and DoS attacks
- Miners choose transactions based on GAS_PRICE
- Overpriced opcodes might be preferred
- Maximum GAS_LIMIT per block

Like in Bitcoin

- All miners needs to
  - Execute all transactions
  - Store code

Unlike Bitcoin

- GAS_LIMIT may halt execution abruptly

Gas costs: https://docs.google.com/spreadsheets/d/1m89CVujrQe5LAFJ8-YAUCcNK950dUzMQPMJBxRtGCqs

1.0 gas costs - Google Sheets

Secure | https://docs.google.com/spreadsheets/d/1m89CVujrQe5LAFJ8-YAUCcNK950dUzMQPMJBxRtGCqs/edit#gid=0

1.0 gas costs

File Edit View Insert Format Data Tools Add-ons Help

arthur.gervais@chainsolutions.com

Share

100% View only

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1		Approximations						Gas price						
2	Param	Compute (µs)	History (bytes)	State (bytes)	Bandwidth	Bloom topic	Mem quad	Computed	**Actual**					Coefficient
3	DUP	3						3	**3**	FASTESTSTEP				
4	SWAP	3						3	**3**	FASTESTSTEP				
5	PUSH	3						3	**3**	FASTESTSTEP		Max execution time (us)	3141592	
6												Max history growth per day (MB)	3217	6.7(
7	ADD	3						3	**3**	FASTESTSTEP		Max state growth per day (MB)	113	190
8	MUL	5						5	**5**	FASTSTEP		Max block size	50.25	6
9	SUB	3						3	**3**	FASTESTSTEP		Max bloom topics per block	12566	250
10	DIV	5						5	**5**	FASTSTEP		Max memory (MB)	39	0.00196
11	SDIV	5						5	**5**	FASTSTEP				
12	MOD	5						5	**5**	FASTSTEP				
13	SMOD	5						5	**5**	FASTSTEP		Gas limit	3141592	
14	ADDMOD	8						8	**8**	MIDSTEP				
15	MULMOD	8						8	**8**	MIDSTEP				
16	EXPBASE	10						10	**10**	SLOWSTEP				
17	EXPBYTE	10						10	**10**					
18	SIGNEXTEND	5						5	**5**	FASTSTEP				
19	LT	3						3	**3**	FASTESTSTEP				
20	GT	3						3	**3**	FASTESTSTEP				
21	SLT	3						3	**3**	FASTESTSTEP				
22	SGT	3						3	**3**	FASTESTSTEP				
23	EQ	3						3	**3**	FASTESTSTEP				
24	ISZERO	3						3	**3**	FASTESTSTEP				
25	AND	3						3	**3**	FASTESTSTEP				
26	OR	3						3	**3**	FASTESTSTEP				
27	XOR	3						3	**3**	FASTESTSTEP				
28	NOT	3						3	**3**	FASTESTSTEP				
29	BYTE	3						3	**3**	FASTESTSTEP				
30	SHA3BASE	30						30	**30**					
31	SHA3WORD	6						6	**6**					
32	ECRECOVER	3000						3000	**3000**					

Sheet1 Sheet2

Gas costs: https://docs.google.com/spreadsheets/d/1m89CVujrQe5LAFJ8-YAUCcNK950dUzMQPMJBxRtGCqs

Transaction Gas

Transaction creator specifies

from	sig	nonce	to	data	value	**gaslimit**	**gasprice**

If GAS_LIMIT x GAS_PRICE > accounts[from].balance, halt

Update the balance

accounts[from].balance -= value + gas x gasprice
accounts[to].balance += value
execute(code)
accounts[from] += unusedGas x gasprice

Gas costs: https://docs.google.com/spreadsheets/d/1m89CVujrQe5LAFJ8-YAUCcNK950dUzMQPMJBxRtGCqs

Out-of-gas Exception

If remaining GAS is 0 before termination, throw out-of-gas

- State reverts to previous state
- GAS_LIMIT x GAS_PRICE is deducted to pay miners

Gas costs: https://docs.google.com/spreadsheets/d/1m89CVujrQe5LAFJ8-YAUCcNK950dUzMQPMJBxRtGCqs

Caller determines the sent gas

Gas costs: https://docs.google.com/spreadsheets/d/1m89CVujrQe5LAFJ8-YAUCcNK950dUzMQPMJBxRtGCqs

Caller determines the sent gas

```
Contract A {
function a():
    assert(msg.gas == 100);
    x = B.b.gas(10)()
    return x + " World!"
```

Gas costs: https://docs.google.com/spreadsheets/d/1m89CVujrQe5LAFJ8-YAUCcNK950dUzMQPMJBxRtGCqs

Caller determines the sent gas

100 gas

```
    Contract A {
→   function a():
        assert(msg.gas == 100);
        x = B.b.gas(10)()
        return x + " World!"
```

Gas costs: https://docs.google.com/spreadsheets/d/1m89CVujrQe5LAFJ8-YAUCcNK950dUzMQPMJBxRtGCqs

Caller determines the sent gas

100 gas

```
Contract A {
function a():
    assert(msg.gas == 100);
    x = B.b.gas(10)()
    return x + " World!"
```

```
Contract B {
function b():
    assert msg.gas == 10
    y = C.c.gas(5)()

  assert(y == 0);
   // out of gas
  return "Hello"
```

Gas costs: https://docs.google.com/spreadsheets/d/1m89CVujrQe5LAFJ8-YAUCcNK950dUzMQPMJBxRtGCqs

Caller determines the sent gas

100 gas

10 gas

```
Contract A {
function a():
    assert(msg.gas == 100);
    x = B.b.gas(10)()
    return x + " World!"
```

```
Contract B {
function b():
    assert msg.gas == 10
    y = C.c.gas(5)()

assert(y == 0);
 // out of gas
return "Hello"
```

Gas costs: https://docs.google.com/spreadsheets/d/1m89CVujrQe5LAFJ8-YAUCcNK950dUzMQPMJBxRtGCqs

Caller determines the sent gas

100 gas

```
Contract A {
function a():
    assert(msg.gas == 100);
    x = B.b.gas(10)()
    return x + " World!"
```

10 gas

```
Contract B {
function b():
    assert msg.gas == 10
    y = C.c.gas(5)()

assert(y == 0);
 // out of gas
return "Hello"
```

```
Contract C {
function c():
 assert(msg.gas == 5);
   while (true) {
       Loop
   }
 return "Bonjour"
```

Gas costs: https://docs.google.com/spreadsheets/d/1m89CVujrQe5LAFJ8-YAUCcNK950dUzMQPMJBxRtGCqs

Caller determines the sent gas

100 gas

```
Contract A {
function a():
    assert(msg.gas == 100);
    x = B.b.gas(10)()
    return x + " World!"
```

10 gas

```
Contract B {
function b():
    assert msg.gas == 10
    y = C.c.gas(5)()

assert(y == 0);
 // out of gas
return "Hello"
```

5 gas

```
Contract C {
function c():
 assert(msg.gas == 5);
   while (true) {
       Loop
   }
 return "Bonjour"
```

Gas costs: https://docs.google.com/spreadsheets/d/1m89CVujrQe5LAFJ8-YAUCcNK950dUzMQPMJBxRtGCqs

Caller determines the sent gas

100 gas

```
Contract A {
function a():
    assert(msg.gas == 100);
    x = B.b.gas(10)()
    return x + " World!"
```

10 gas

```
Contract B {
function b():
    assert msg.gas == 10
    y = C.c.gas(5)()

assert(y == 0);
 // out of gas
return "Hello"
```

5 gas

Out of Gas Exception

```
Contract C {
function c():
 assert(msg.gas == 5);
   while (true) {
       Loop
   }
 return "Bonjour"
```

Gas costs: https://docs.google.com/spreadsheets/d/1m89CVujrQe5LAFJ8-YAUCcNK950dUzMQPMJBxRtGCqs

Caller determines the sent gas

100 gas

```
Contract A {
function a():
    assert(msg.gas == 100);
    x = B.b.gas(10)()
    return x + " World!"
```

10 gas

```
Contract B {
function b():
    assert msg.gas == 10
    y = C.c.gas(5)()

assert(y == 0);
 // out of gas
return "Hello"
```

5 gas

```
Contract C {
function c():
 assert(msg.gas == 5);
   while (true) {
       Loop
   }
 return "Bonjour"
```

returns “Hello World”

Out of Gas Exception

Gas costs: https://docs.google.com/spreadsheets/d/1m89CVujrQe5LAFJ8-YAUCcNK950dUzMQPMJBxRtGCqs

How do you think miners order transactions in a block?