By 区块链 keep 哥

结论：

a. 目前 EOS 的 Ram 大小是写死在程序里的，所以不升级软件是绝对不可能扩容的，市场上各种扩容的风传完全站不住脚

b. Bancor relay weight 的从 0.5 变化 50 不会改变 ram 的价格趋势，计算过程我上传到 https://pan.baidu.com/s/1U3LCAVDUT5lHHdGPGDp0bw。 有兴趣的同学可以下载检验一下，如果你觉得有计算过程有疑问可以和我联系。

c. Ram 占用率 vs 价格趋势如下，占用率达到 87.78%时，ram 的价格是 1eos/Kbyte；当占用达到 90.01%时，ram 的价格是 1.5 eos/Kbyte.

Bancor weight = 0.5时， ram 占用率 vs 价格趋势	
行标签	求和项:平均价格(Eos/KB)
80.00%	0.373641016
85.01%	0.665573182
87.78%	1.001819136
90.00%	1.497536378
90.01%	1.500518026

Bancor weight = 50时， ram 占用率 vs 价格趋势	
行标签	求和项:平均价格(Eos/KB)
80.00%	0.373641016
85.01%	0.665573182
87.78%	1.001819136
90.00%	1.497536378
90.01%	1.500518026

详细分析：

1. 目前 EOS 的 Ram 大小是写死在程序里的，所以不升级软件是绝对不可能扩容的，市场上各种扩容的风传完全站不住脚

```
struct eosio_global_state : eosio::blockchain_parameters {
   uint64_t free_ram()const { return max_ram_size - total_ram_bytes_reserved;

   uint64_t             max_ram_size = 6411*1024 * 1024 * 1024;
   uint64_t             total_ram_bytes_reserved = 0;
   int64_t              total_ram_stake = 0;

   block_timestamp      last_producer_schedule_update;
   uint64_t             last_pervote_bucket_fill = 0;
   int64_t              pervote_bucket = 0;
   int64_t              perblock_bucket = 0;
   uint32_t             total_unpaid_blocks = 0; /// all blocks which have been produced but not paid
   int64_t              total_activated_stake = 0;
   uint64_t             thresh_activated_stake_time = 0;
   uint16_t             last_producer_schedule_size = 0;
   double               total_producer_vote_weight = 0; /// the sum of all producer votes
   block_timestamp      last_name_close;

   // explicit serialization macro is not necessary, used here only to improve compilation time
   EOSLIB_SERIALIZE_DERIVED( eosio_global_state, eosio::blockchain_parameters,
                             (max_ram_size)(total_ram_bytes_reserved)(total_ram_stake)
                             (last_producer_schedule_update)(last_pervote_bucket_fill)
                             (pervote_bucket)(perblock_bucket)(total_unpaid_blocks)(total_activated_stake)(thresh_activated_stake_time)
                             (last_producer_schedule_size)(total_producer_vote_weight)(last_name_close) )
```

Ram 的总量是是写死在程序里的，所以不升级软件是绝对不可能扩容的，所以市场上各种扩容的风传完全站不住脚。

如果要按照 BM 说的，每产生一个区块，扩容一点点，这个代码的逻辑会很复杂，正常情况下，短期(一周)内没有上线的可能性。

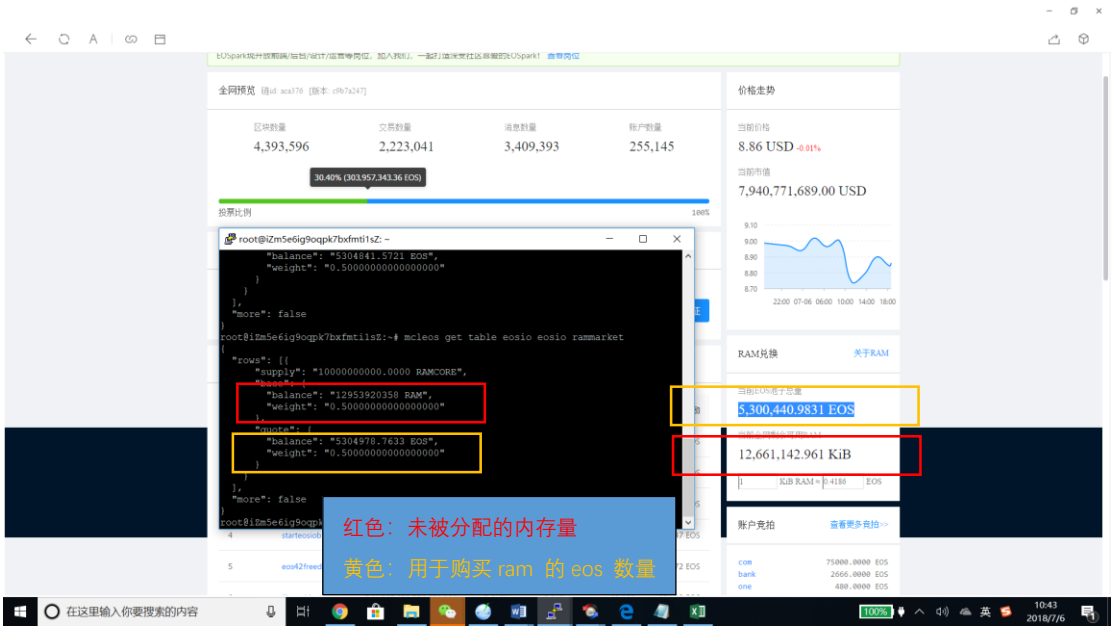
如果各位写过程序，对于这种写法，是不是想开骂？😉。

2. Eos 使用 rammarket 来记录可用内存量和货币量的关系

在 eosio.system 合约下有一张表 rammarket，它用来记录内存量和货币量的比值。

使用 cleos get table eosio eosio rammarket，可以看到其中的数据。

在 2018/7/6 10:43 分查询，可以看到：base.balance 记录了当前未被购买的 ram 大约有 12G；用于购买 ram 的 eos 大约有 530 万。



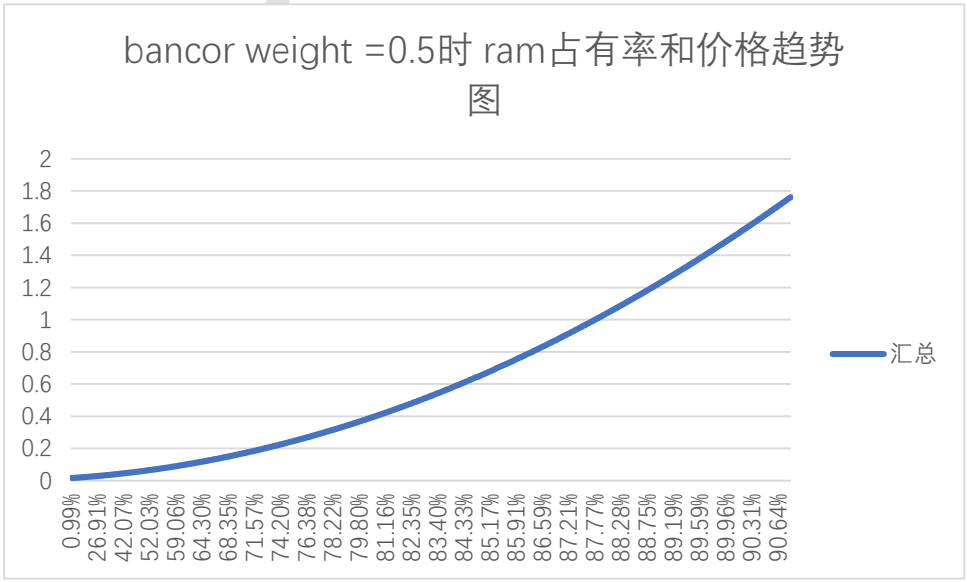

```
root@iZm5e6ig9oqpk7bxfmti1sZ:~# cleos -u http://mainnet.genereos.io get table eosio eosio rammarket
{
  "rows": [{
      "supply": "10000000000.0000 RAMCORE",
      "base": {
        "balance": "12979869510 RAM",
        "weight": "0.50000000000000000"
      },
      "quote": {
        "balance": "5294373.1753 EOS",
        "weight": "0.50000000000000000"
      }
    }
  ],
  "more": false
}
```

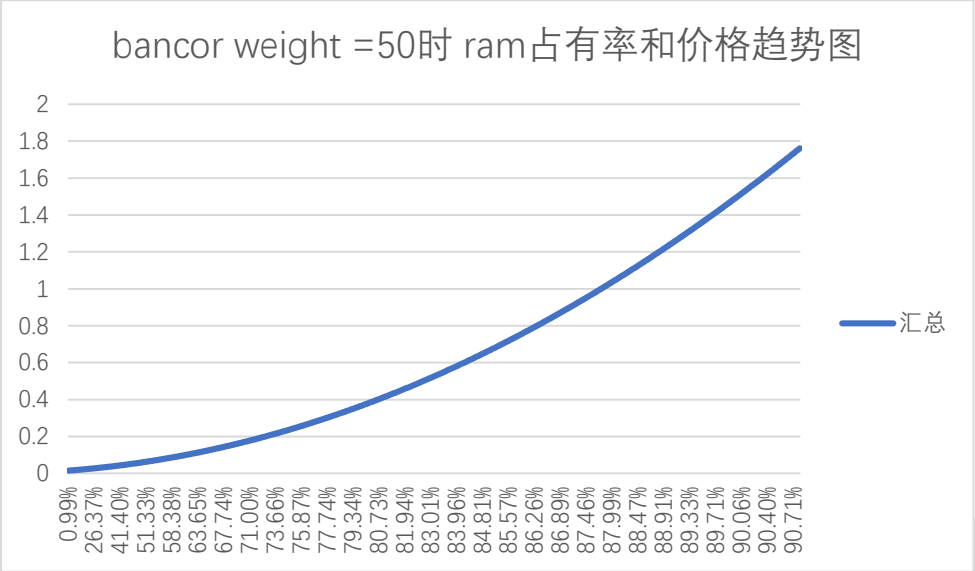
rammarket 是在首次调用 eos 系统自带的智能合约，初始化时会将可用内存(64G)，以及购买内存的 eos 记录下来,需要说明一点，按照代码逻辑，初始化的时候有 100 万的 eos 用于购买 0KB 的内存。

```
system_contract::system_contract( account_name s )
:native(s),
 _voters(_self,_self),
 _producers(_self,_self),
 _global(_self,_self),
 _rammarket(_self,_self)
{
   //print( "construct system\n" );
   _gstate = _global.exists() ? _global.get() : get_default_parameters();

   auto itr = _rammarket.find(S(4,RAMCORE));

   if( itr == _rammarket.end() ) {
      auto system_token_supply   = eosio::token(N(eosio.token)).get_supply(eosio::symbol_type(system_token_symbol).name()).amount;
      if( system_token_supply > 0 ) {
         itr = _rammarket.emplace( _self, [&]( auto& m ) {
            m.supply.amount = 100000000000000ll;  //初始化supply，这是Ram 和EOS 换算的一个中间量，100亿,小数点后有4位。
            m.supply.symbol = S(4,RAMCORE);
            m.base.balance.amount = int64_t(_gstate.free_ram()); //base.amount 记录了目前还未分配的内存量，EOS启动的那一刻是64G.
            m.base.balance.symbol = S(0,RAM);
            m.quote.balance.amount = system_token_supply / 1000; // 100万 eos
            m.quote.balance.symbol = CORE_SYMBOL;
         });
      }
   } else {
      //print( "ram market already created" );
   }
} // end system_contract
```

调用 buyram 函数来计算 quant 个 eos，能卖多少 byte 的 ram.

3. **Covert 函数实现了 bancor 算法，分析的结果是 bancor relay weight 的从 0.5 到 50 不会改变价格趋势。**




bancor weight =50时 ram占有率和价格趋势图
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0.99%
26.37%
41.40%
51.33%
58.38%
63.65%
67.74%
71.00%
73.66%
75.87%
77.74%
79.34%
80.73%
81.94%
83.01%
83.96%
84.81%
85.57%
86.26%
86.89%
87.46%
87.99%
88.47%
88.91%
89.33%
89.71%
90.06%
90.40%
90.71%
汇总

By Keep