Tail chain - a new generation blockchain for parallel processing

Mikhail P. Levin, Nikolay Pakulin, Andrey Tapekhin

Pax Datatech & Ivannikov Institute of System Programming RAS,

Mikhail_Levin@hotmail.com; Mikhail_Levin@paxdatatech.com

Nikolay@paxdatatech.com, a.tapekhin@ispras.ru

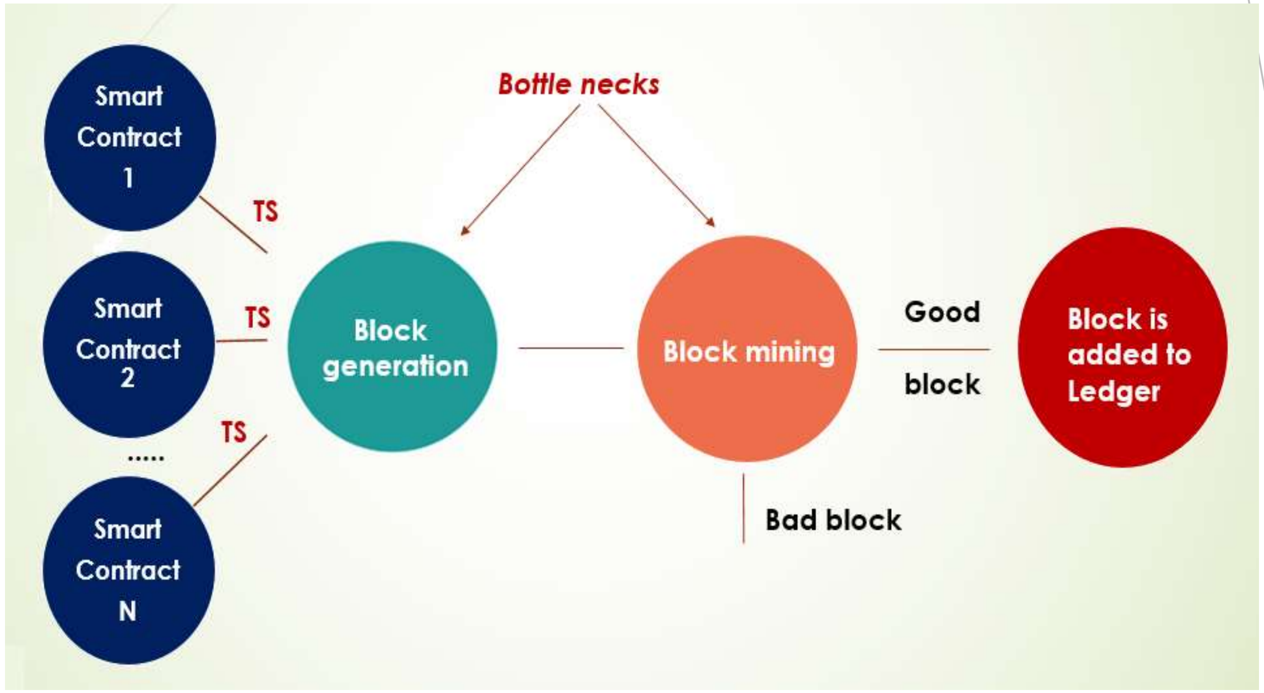
Introduction

- Starting from 2008 blockchain technologies are widely and intensive developed over the world
- Now a days there are a lot of various variants of this technology were developed. But in principle all these variants are concerned around Bitcoin and Ethereum platforms
- In these platforms transactions flow comprises the following steps:
  a) New transactions are propagated and advertised to all nodes
  b) Every miner node collects new transactions into a block
  c) Nodes look for proof of work
  d) When a node finds a proof of work, it broadcast the block to all other nodes
  e) Nodes accept the block only if all transactions in it are valid and not already spent
  f) Nodes accept the block by working on creative the next block in the chain, using hash of the accepted block as previous hash

State of Art

In general, transaction flow processes can be presented by the following scheme

- Blockchain clients create contracts and generate transactions TS
- These transactions are collected by any specified miner node into the block of transaction (see: Block generation element)
- When the block of transaction is completed, it is sent to any selected set of miners for proofing (see: Block mining element)



- In the last operation should be two cases.
  - One is: the block is good, then this block is added to the blockchain.
  - Another is: the block is bad, in this it drop out and does not add to the blockchain

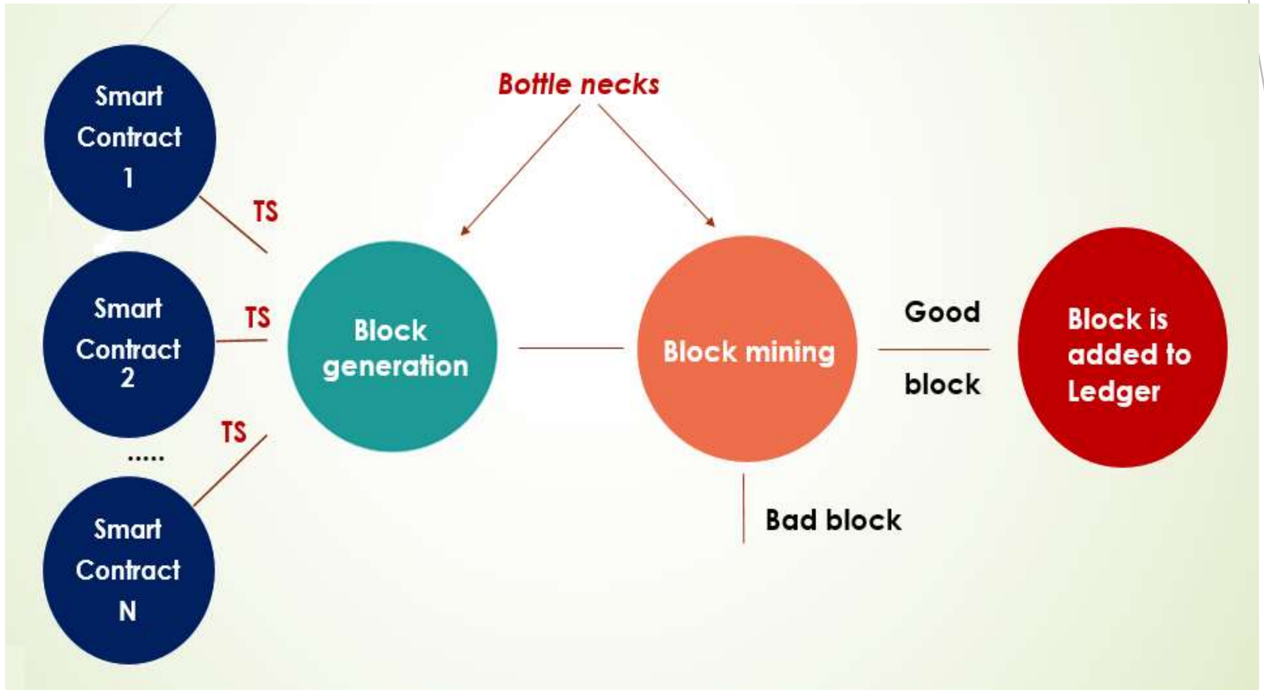
Bottle necks: only one node at a time constructs a block (both in PoW and PoS)

$$performance = \frac{tnx\_per\_block}{t_{mining} + t_{consensus}}$$

State of Art

In general, transaction flow processes can be presented by the following scheme

- Blockchain clients create contracts and generate transactions TS
- These transactions are collected by any specified miner node into the block of transaction (see: Block generation element)
- When the block of transaction is completed, it is sent to any selected set of miners for proofing (see: Block mining element)



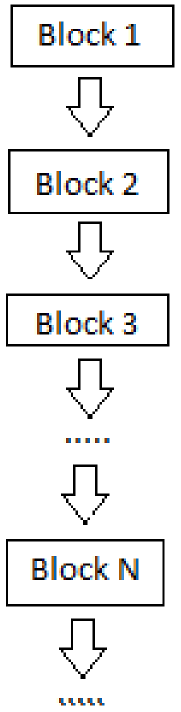
- In the last operation should be two cases.
  - One is: the block is good, then this block is added to the blockchain.
  - Another is: the block is bad, in this it drop out and does not add to the blockchain

Bottle necks: only one node at a time constructs a block (both in PoW and PoS)

$$performance = \frac{tnx\_per\_block}{t_{mining} + t_{consensus}}$$

State of Art

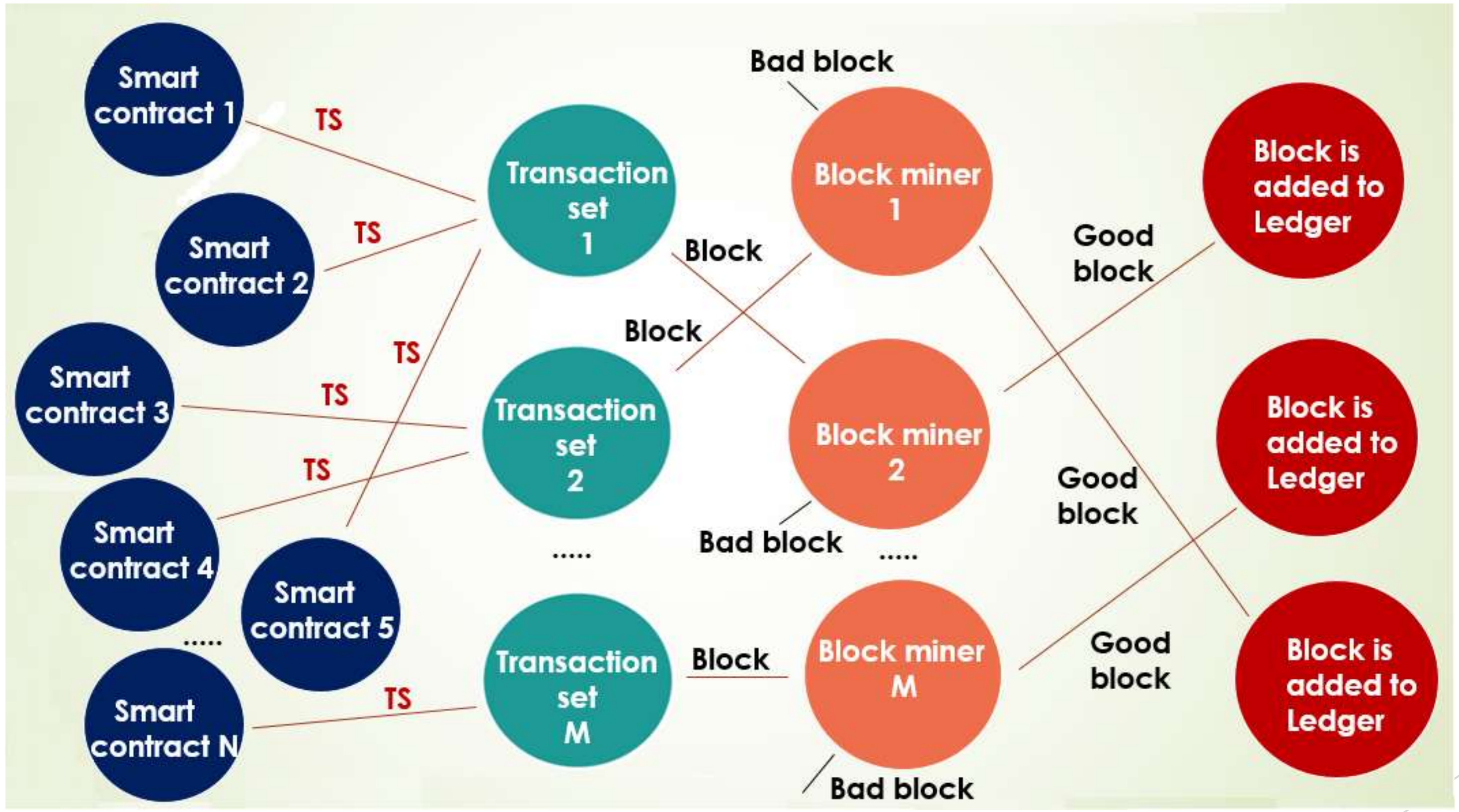
- Structure of blocks in nowadays blockchain
- Two bottlenecks:
  - one is when the current block is generated, because before the block is not completed we could not send it to the next stage
  - The second bottleneck is on the mining stag this stage a few mines should take part in th block proofing with respect to the choosing algorithm of proofing (Proof of Work, Proof Stake (PoS), Proof of Elapsed Time and some others)
- All proofing results of miners participated i proofing are accumulated on one miner no there the final solution is done
- These two above mentioned bottlenecks seriously restricted performance of mining process



$$performance = \frac{tnx\_per\_block}{t_{mining} + t_{consensus}}$$

Tail chain idea

- Transaction scheduling is performed by distributed process
  Add a new role - Transaction Dispatcher
  Transaction Dispatcher forms a transaction set for Block Generation
  Each node in the network might be both Block Generator (Miner or Minter) and Transaction Dispatcher
- After forming a transaction set Transaction Dispatcher is re-elected by a PoS algorithm (or other pseudo-random selection strategy)
- Formed transaction set is distributed as a message to all nodes in the network
- The next Transaction Dispatcher forms a set that has zero intersection with this one (each transaction must appear in a single transaction set)



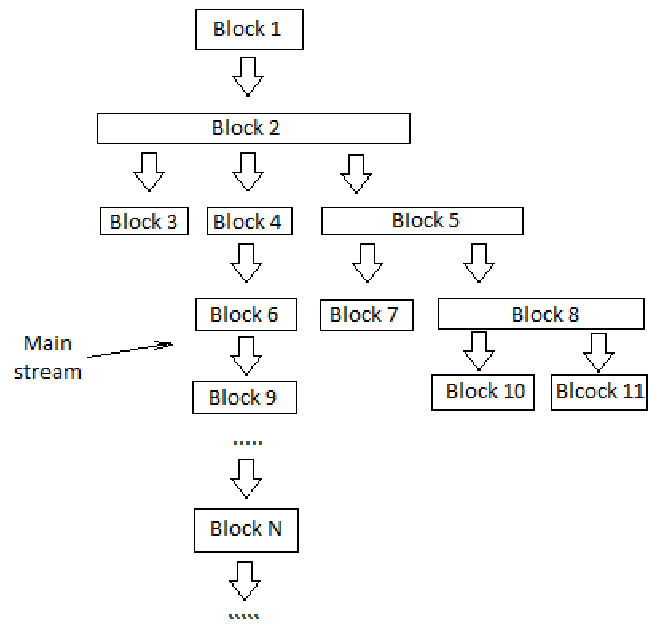
- When Block Generator (selected by PoS algorithm) receives a transaction set message
- it mints the transactions from the set into a block
  - Forming Transaction Set is much faster than generating a block
  - Therefore multiple transaction sets could be minted at the same time

Parallel minting !

Structure of Tail chain

- Tail chain graph always has a main stream and we open a new stream only then a main stream is busy
- Then the main stream is not busy and waiting a new block, then we closed all peripheral streams in Tail chain graph. By such way we avoid an unlimited growth of streams in Tail chain graph
- We should define a maximal number of streams in Tail chain graph and close all peripheral streams then this number is achieved (this of course can decrease performance just a little)
- Let us note that the proposed process requires Proof-of-Stake or similar validation algorithm and does not work with Proof-of-Work.

Structure of blocks in Tail chain



Performance Estimation

$$performance = \frac{tnx\_per\_block}{t_{mining} + t_{consensus}} \times num\_of\_nodes$$

- Almost unlimited scaling
- Requires Proof-of-Stake or similar
  - Does not work with Proof-of-Work

This work was supported by the Minobrnauki Russia (grant id RFMEFI60417X0199, grant number 14.604.21.0199)

Thanks for attention!