Blockchain as a Database: Limitations and Opportunities

SEMINAR @ SOFTUNI
26.11.2017

SPEAKER: HRISTO HRISTOV
BUSINESS ANALYST, MCP, CBP

WWW.CHRISSTOV.COM

Agenda

1. **Seminar opening**
2. **Introduction**
   - Relational / NoSQL database vs. Blockchain
3. **Blockchain data storage**
   - 3.1. Data storage on the blockchain – smart contracts
   - 3.2. Data storage on the blockchain – transactions
   - 3.3. Limitations
4. **Blockchain as a Database**
   - 4.1. MSSQL and blockchain integration
   - 4.2. MongoDB and blockchain – blockchainification*
   - 4.3. Overall blockchain application in (big) data
5. **Use case**
6. **Q & A**

2. Introduction

Relational (SQL) database	Blockchain	Note
Tables (attribute-value pairs), have unlimited and cheap data storage	Blocks of data, limited and expensive data storage	Prepare datasets and deploy them on the blockchain
Fast CRUD	CRUD can be slow, expensive, depends on consensus algorithm, block propagation	An all-accessible source of information with instant access
Data replication can be slow	Inherent data replication, every node has a copy	Do you need it?
Update or delete at any time	Tamper-proof, immutable	Need to add information to another block, with the previous information still there

2. Introduction – another perspective

	Characteristic	Database	Blockchain
Database characteristics	Queryability	?	
	Scale	?	
	Operational	?	
Blockchain characteristics	Decentralized		?
	Immutable		?
	Assets		?

Gartner: Blockchain Tech Hits Hype Cycle Peak: https://goo.gl/Z72Lc3

2. Introduction - blockchain

- Decentralized database
- Peer-to-peer vs. client-server
- How can you store a whole DB in each block?
  - Throughput is just a few transactions per second (tps),
  - Latency before a single confirmed write is ~10 minutes,
  - Capacity is a few dozen GB
  - Concurrency? – not really
  - Queryability? – APIs

3.1. Smart contract data storage

3.1.1. Ethereum

- Prohibitive and expensive
- No direct limit
- Block gas limit
- No standard way to **SELECT** data
- Contract calls to functions will read the data
- Query: call a function via an interface

3.1. Smart contract data storage

3.1.2. NEO

- Storage.Put() [per KB] – 1 GAS (~$20/GAS)
- Storage Class - Provides a set of methods to insert, query, and delete data in the persistent store (static).

3.2. Data storage in transactions

3.2.1. Ethereum

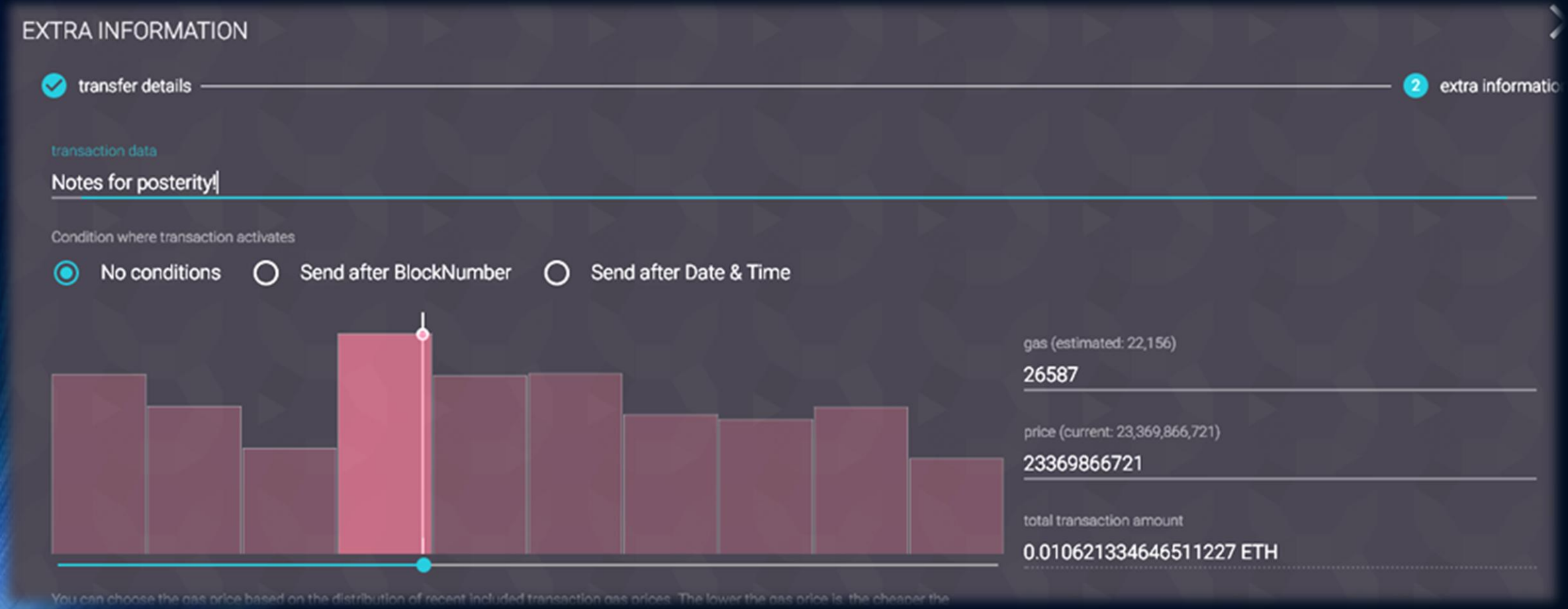
- Transaction input
- Arbitrary string
- No explicit limit - only block gas limit
- No limit in private blockchains
- Anyone can look up the data
- Better to use hash or encryption
- No direct way to SELECT data
- All sorts of combinations thereof, e.g. hash a whole database and upload the hash to an Ethereum block
- Query: JSON interface, .e.g. etherscan.io, design a way to query data – from…to, address, contains…

3.2. Data storage in transactions

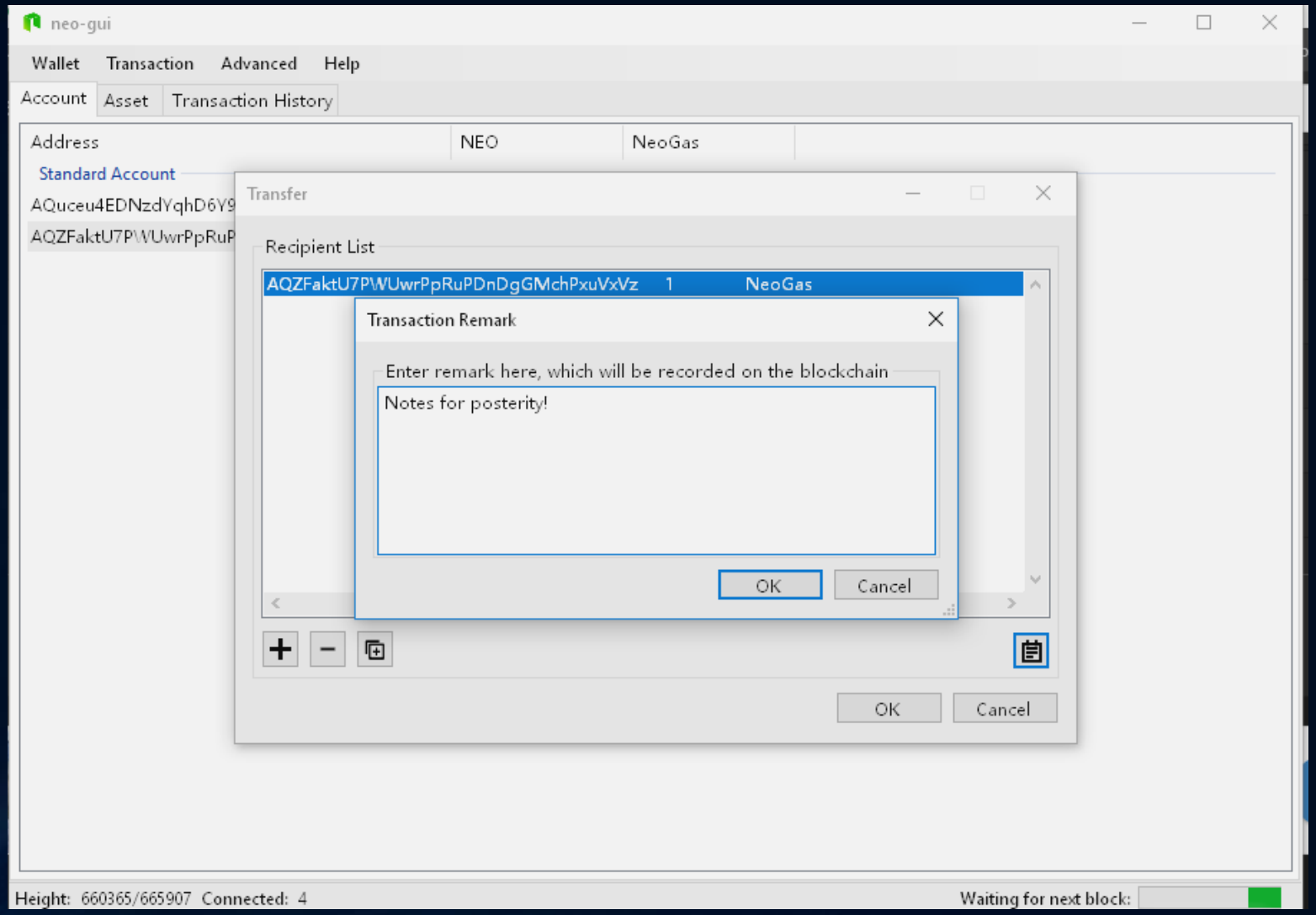
3.2.2. Transaction remark – NEO

- user-unfriendly

Parity note interface



NEO wallet – note interface



3.3. Limitations and Opportunities

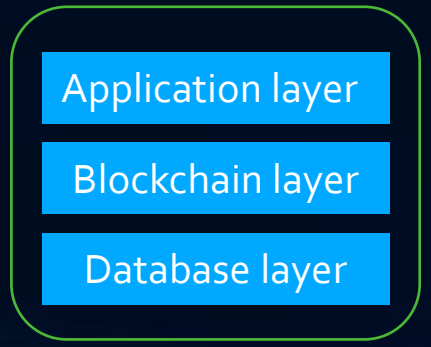
- Blockchain cannot be used directly for storing any other data except for financial transactions
- Integration is the way to go
- Mapping between
  - Small scale: database entities and block data
  - Large scale: data artifacts or model and block data

4.1. MSSQL and Blockchain

- Most direct way of "integration" – store the database logs
- Combine access to DB with a smart contract
- Enable data retrieval with smart contract/s
- Enterprise – CoCo Framework (Confidential Consortium)
  - 1600 tx/s
  - Consortium model
  - Nodes and actors can be controlled

4.2. MongoDB and blockchain

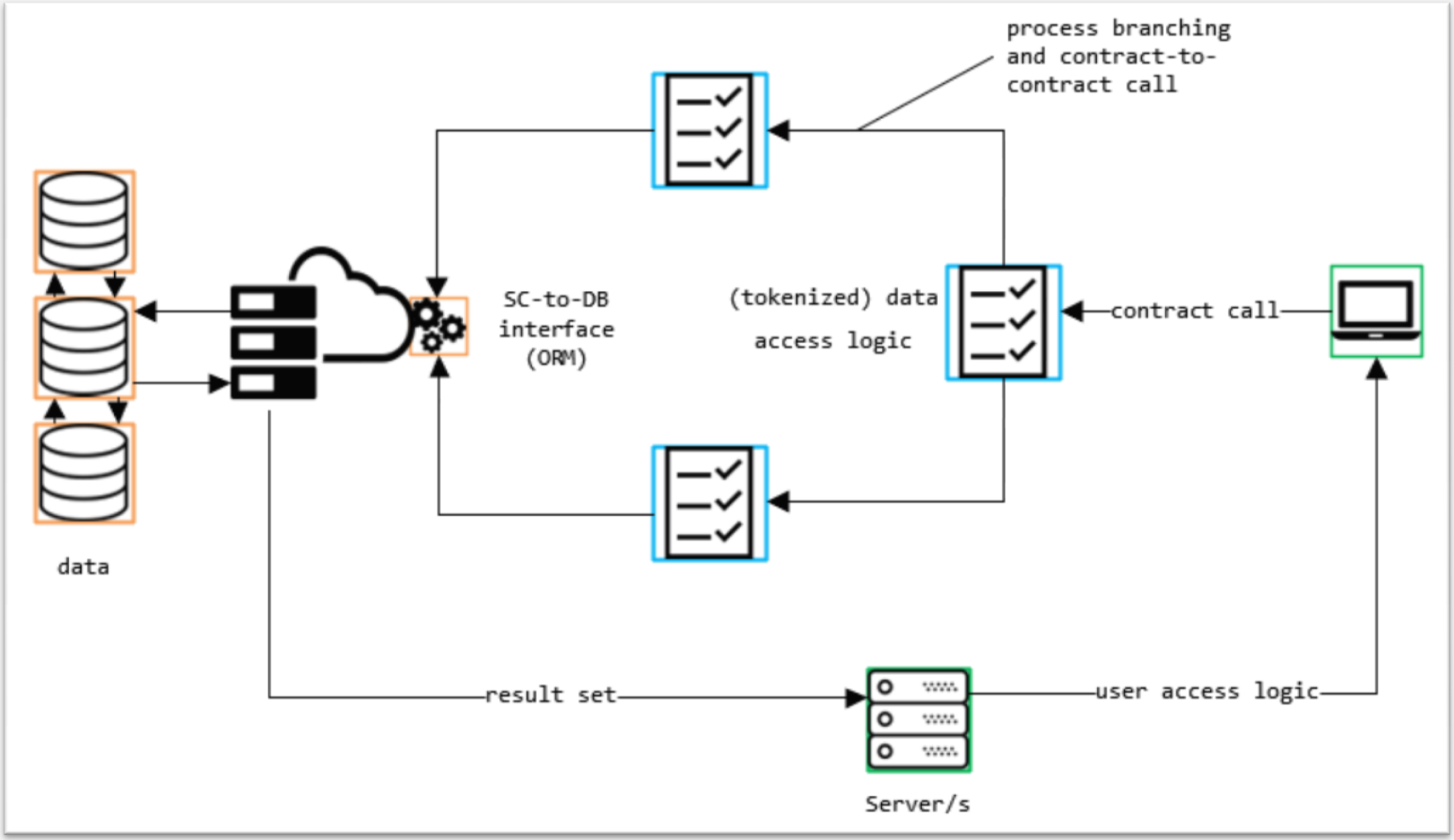
- Blockchainified MongoDB
- Operational data – what needs to be exposed to internal or external parties **from the blockchain**
- Store block data or metadata in the database



4.3. Overall blockchain applications in data

- Ensure data integrity and consistency
- Increase confidence in data
- Help answer the following questions:
  - If you are a multinational corporation, how do you share data around the planet?
  - How do you assign access rights based on business logic?
  - If you generate the data how do you prove it was you?
  - How do you transfer the rights of the data to others?
  - How do you ensure a next-generation audit trail?

5. High level overview - data access model



- a data set (query) is being requested via the smart contract (open methods)
- tokenized access possible
- token issuance upon read request possible
- pay for delivering information in various formats: Excel, CSV, JSON/BSON, XML, SQL
- data is being served outside of the blockchain
- this is how you create data driven eco-systems
- scheduled / event-based / time-based access

Thank you!

Questions / comments?

linkedin.com/in/hthristov