A New Smart Contract Solution

Designed by: Tri Nguyen-Pham
Written by: Michael Rainwater

Problem: New smart contract solutions face a myriad of challenges that must be met with new developments capable of solving the first generation's problems. Several major problems exist with current smart contract platforms. Platforms like Ethereum and Tezos utilize uncommon programming languages such as Solidity, Vyper, and Mycelium. On Ethereum, smart contracts are limited in their scalability due to their existence in the Ethereum blocks which have a hard coded size limit. Furthermore, only one smart contract execution method can be submitted at a time.

Solution: Raptoreum aims to solve these problems with a new approach to decentralized program storage and execution within blockchain technology. Through a combination of the Raptoreum Smartnode network and Apache Spark™, the second generation of smart contract solutions can be developed and executed. Although the initial setup transaction for every Smartnode is secured through transactions on the Raptoreum blockchain, the Smartnode layer itself exists outside of the chain on CPUs around the world. Raptoreum will initially utilize these machines for the purposes of hosting and executing smart contracts using commonly known programming languages interoperably with the help of Apache Spark™. The actual work processes will be organized and distributed via Apache Spark™ to provide a platform that "achieves high performance for both batch and streaming data using a state-of-the-art DAG scheduler, a query optimizer, and a physical execution engine (1)." Raptoreum will support batch job submissions as well, improving network throughput as developers will be able to execute batches of smart contract requests all at once.

I. Smartnodes

Raptoreum Smartnodes are the next step in an evolution which began with Dash masternodes. Dash explains that “Masternodes host full copies of the blockchain and provide a unique second layer of services to the network, facilitating advanced functions such as InstantSend, PrivateSend and usernames on the blockchain (2).” Raptoreum takes this concept a step further by giving masternodes the ability to store and execute smart contracts that operate more freely through a decentralized processing network as opposed to being confined to the inside of the blockchain itself. This is where the term “Smartnode" was derived from.

When it comes to the Raptoreum network, Smartnodes perform a variety of vital functions that allow for a secure, fast, and developable network. Unlike other Masternode based cryptocurrency networks, Raptoreum will run smart contracts on the Smartnode layer to avoid the first generation limitations which arise from running contracts inside the blocks of the ledger. RTM Smartnodes utilize DASH-style chain locks to prevent the ledger from being reorganized further than 1 block. This means that Raptoreum is immune to the traditional 51% attack because an attacker would not only need to control 51% of the overall network hashrate, but they would also need to control over 60% of the active Smartnodes on the network simultaneously. This makes such an attack virtually impossible and extremely expensive.

“If a valid CLSIG message is received by a node, it should reject all blocks (and their descendants) at the same height that do not match the block specified in the CLSIG message. This makes the decision on the active chain quick, easy and unambiguous. It also makes reorganizations below this block impossible.(3)*”*

With a projected network target of around 4-6000 Smartnodes, each equipped with a minimum of 8 core CPUs with 16Gb RAM and 512 GB SSD, there will be ample distribution of available resources to provide a fast, powerful, and decentralized smart contract platform that is more versatile in it’s programmability than any of its

predecessors. This layer will house the binary code for smart contracts and other future decentralized applications, which will always be verifiable through a matching binary hash stored on the Raptoreum blockchain when the contract or application is launched. Since Smartnode chain locks prevent reorganization of the RTM blockchain, they also protect all binary hash transactions from loss or tampering, thereby providing immutable verification of the matching binary code for as long as the network is running.

II. Smart Contracts

IBM explains that smart contracts are:

"programs stored on a blockchain that run when predetermined conditions are met. They typically are used to automate the execution of an agreement so that all participants can be immediately certain of the outcome, without any intermediary's involvement or time loss. They can also automate a workflow, triggering the next action when conditions are met (4)*."*

When they were first developed, smart contracts allowed for the execution of code upon the blockchain. This represented a big leap forward for blockchain technology and it opened the doors to many new use cases through the development of decentralized applications that are able to exist and execute inside blocks on the blockchain itself. This is how Ethereum and other first generation smart contract platforms currently operate.

Raptoreum will improve upon this technology by moving smart contracts onto the Smartnode network layer and out of the limited space inside of the blockchain itself. This allows for a much greater degree of depth and flexibility when it comes to programming compared to previous smart contracts. With greater flexibility of programming, maximum data size, and multiple programming languages, many of the limits that restrict developers on current smart contract platforms are removed if developers choose to utilize the Raptoreum network.

III. Apache Spark™

Apache Spark™ is a commercial-grade unified analytics engine ready for large-scale applications. As such, it is already in use by academic laboratories, Fortune 500 companies (5), and major corporations such as:

- Amazon
- eBay Inc. log transaction aggregation and analytics
- Credit Karma to create personalized experiences
- Groupon
- IBM Almaden
- NASA JPL - Deep Space Network
- Nokia Solutions and Networks
- Shopify
- Stanford DAWN for usable machine learning
- TripAdvisor
- VideoAmp Intelligent video ads for online and television viewing audiences.
- Yahoo!

Source: https://spark.apache.org/powered-by.html

Initially launched by the UC Berkeley AMPLab, "Apache Spark™ is a unified analytics engine for large-scale data processing. Spark offers over 80 high-level operators that make it easy to build parallel apps. And you can use it *interactively* from the Scala, Python, R, and SQL shells(1)." With Spark, it is also possible to write applications, and therefore RTM-based Smart Contracts, in Java.

With Spark running on the Smartnode layer of the Raptoreum network, RTM-based smart contracts in these common programming languages will be immediately accessible to millions of developers around the world. By bringing "Apache Spark™

onto the Rapterium Smartnode network layer, smart contracts will be able to utilize “a stack of libraries including SQL and DataFrames, MLlib for machine learning, GraphX, and Spark Streaming. You can combine these libraries seamlessly in the same application(1).” Deploying smart contracts on the Raptoreum network will allow for the improved functionality of current use cases and for new use cases that were impossible before when using existing smart contract platforms.

IV. The Raptoreum Smart Contract Solution

How it Works with RTM - The exact process that is involved with utilizing smart contracts on the Raptoreum network may change during testing and implementation, but what follows is a simple step-by-step overview of the general process as it has been conceived thus far:

1. A user submits a job or batch of jobs through an execution request by using an RPC call. This creates a special type of transaction on the Raptoreum chain. Once the transaction is confirmed, the job(s) will enter the execution queue and the Smart Contract is secured to the chain.

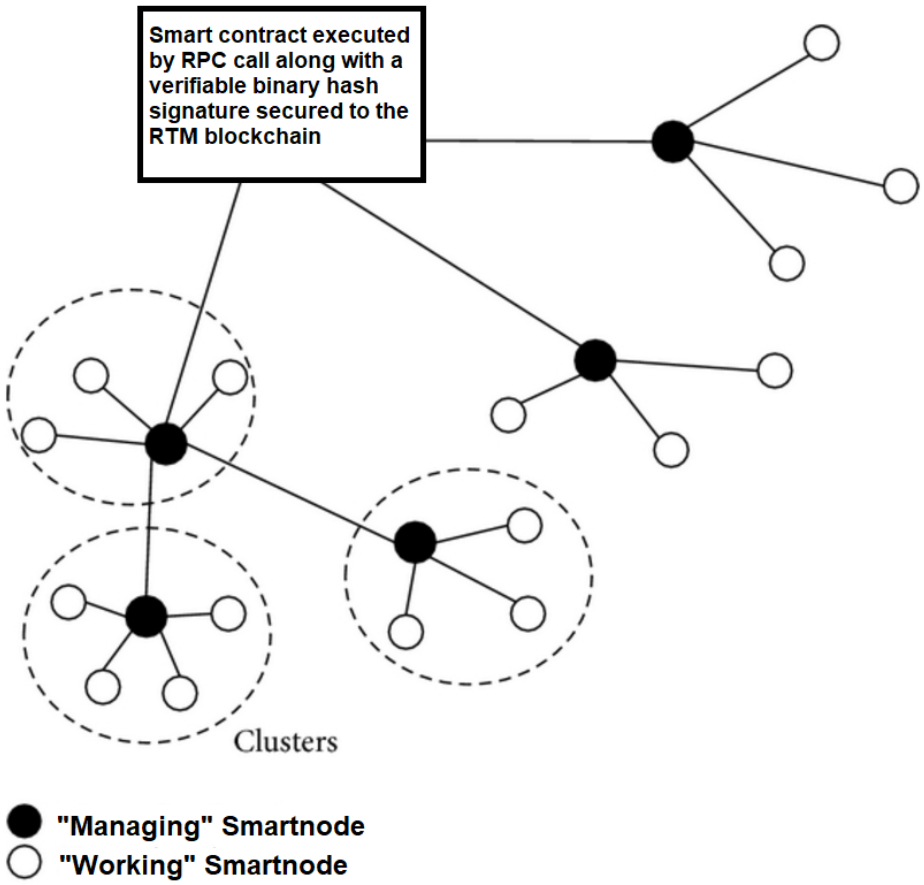
2. The Smartnodes create a quorum with each other to check the binary hash against the original smart contract creation.

3. “Managing” Smartnodes listen to the job queue and one Smartnode is elected to submit the job(s) using Livy to create a spark-submit.

4. Both steps 2 & 3 take place in the same Quorum message.

5. When the Livy spark-submit is successful, the Apache Spark™ system takes over and directs a cluster of “Working” Smartnodes within the Smartnode network to complete the job.

6. Job result (Success/Failure) is broadcast to all nodes and, if successful, the result is recorded on the chain or on a database maintained by the Smartnodes.

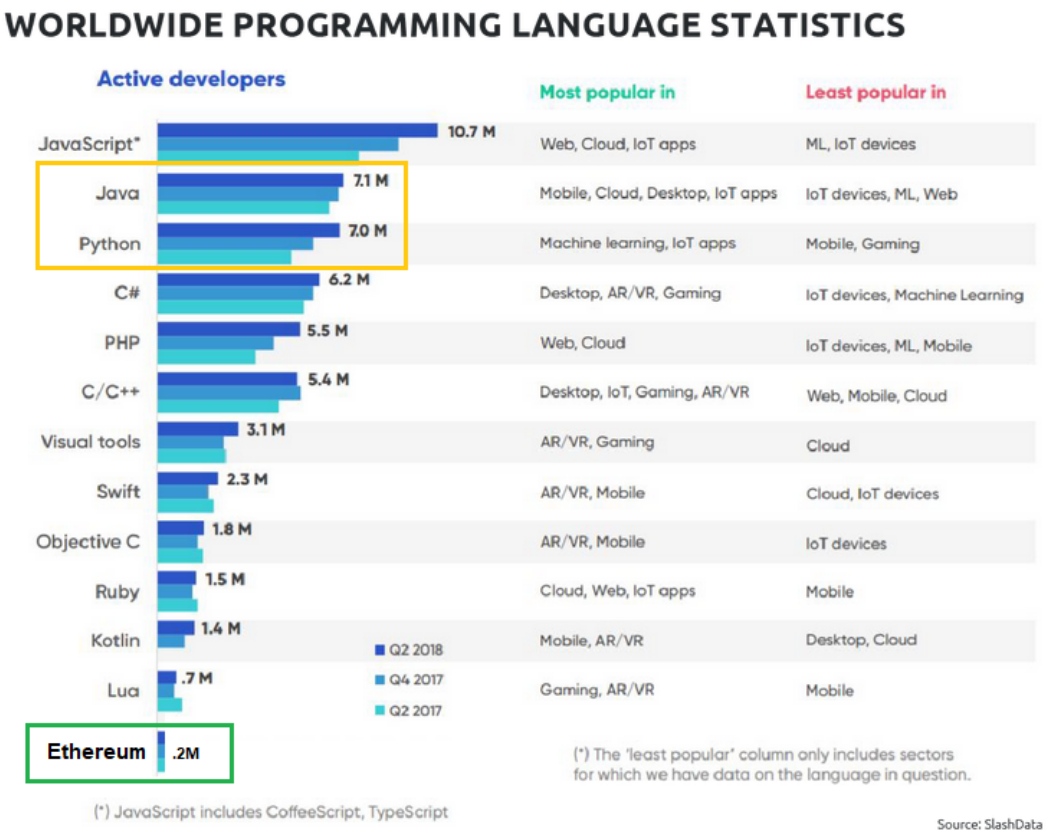


7. Raptoreum full Smartnodes can send zmq messages, which allows any connected application to listen and react in real time based on the results of job submissions.

This network design allows for many exciting new possibilities when it comes to decentralized and automated job execution secured by blockchain technology. One example would be to have a token standard in a specific language that uses a specific spark binary to meet the standardization requirements. This allows for the open sourcing of standardized token creation and on-network verification of

that open source code when a user wishes to create a token with that standard. Raptoreum smart contracts will have far more execution options than Ethereum or any other smart contract platform currently in existence.

Benefits - One of the most immediate and significant benefits to this new Smart Contract solution is the immediate availability of millions of new developers around the world. To put it in perspective, in 2018 Trustnodes.com determined there might be only about “200,000 ETH developers ecosystem wide (6).” There were roughly 7.1 million Java developers and 7 million Python developers alone that same year.



Source: https://www.daxx.com/blog/development-trends/number-software-developers-world

Some additional benefits of utilizing the Raptoreum Smartnode network for smart contract execution include:

- The first ever “value based” checks built into the chain which will vary the cost of smart contract execution services based off of value over time instead of a set number of RTM coins. This will avoid the situation of crippling development due to overly expensive transaction fees

- Timelocks are for auto-contract triggers that can set a future termination time for contracts when they are created (future-based contracts)

- Larger sized smart contracts

- Transaction fees are extremely low compared to other Smartnode platforms

- Job batch submission and execution

- Token standardization with on-chain binary verification

- Real-time communication between decentralized jobs and listening applications running inside or outside of the Raptoreum network

- Decentralized job queue so that reactive Smartnode clusters can pickup jobs from other clusters should Smartnodes go offline while completing jobs for any reason

- Smartnode requirements mean there are enough resources for large scale data processing and job execution for decentralized applications

Risks/Pitfalls -

As with any step taken into unknown territory, there are likely to be a number of risks and potential pitfalls involved in the creation of the Raptoreum smart contract solution. Some known risks that could potentially be faced by the Raptoreum team include:

- Lack of developer support
- Lack of user support
- Bad actors utilizing the platform for malicious contracts

Future Applications -

The future applications of having smart contracts and Apache Spark™ working together on the Smartnode layer of the Raptoreum network will be unlike anything that currently exists today. Smart contracts will initially be able to provide autonomous services and a platform for assets. With the increased versatility of the smartnode network design, future applications will be able to utilize various functions that currently only exist separately in other distinct blockchains. These functions will be able to be performed simultaneously within the Smartnode layer of the Raptoreum network.

By providing this new RTM-based framework for smart contracts, futures, assets, NFTs, and decentralized applications that is faster, more robust, and more programmable than ever before, all-new use cases powered by decentralized applications can emerge that were not possible before on any other smart contract framework. Just about anything that can be programmed using the languages mentioned above can be stored and executed on the Raptoreum network when Apache Spark™ is utilized on the Smartnode Layer.

V. Disclaimer

This document is not financial advice and it contains no financial advice. This is meant to be a fluid document for driving future development. This document is not a guarantee of future development. Nothing in this document is final. Everything in this document is subject to change with or without notice. Future refinement may or may not involve developments that are different than described or altogether not mentioned in this document.

VI. Sources

1. https://spark.apache.org/
2. https://www.dash.org/masternodes/
3.https://blog.dash.org/mitigating-51-attacks-with-llmq-based-chainlocks-7266aa648ec9
4. https://www.ibm.com/topics/smart-contracts
5. https://fortune.com/fortune500/2021/search/
6.https://www.trustnodes.com/2018/07/22/ethereums-ecosystem-estimated-200000-developers-truffle-seeing-80000-downloads-month

Contacts

Discord: https://discord.gg/2T8xG7e

Telegram: https://t.me/raptoreumm

Twitter: https://twitter.com/raptoreum

Reddit: https://www.reddit.com/r/raptoreum