Cytonic: A multi-virtual-machine distributed network

badconfig@cytonic.com
www.cytonic.com

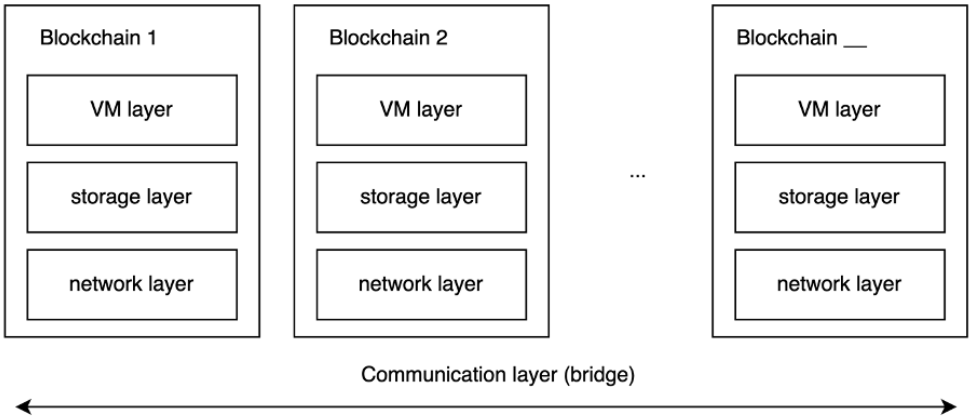
5th November, 2024

Abstract

Achieving compatibility among diverse blockchain ecosystems presents a significant challenge due to differing runtimes and protocols. We propose a method to seamlessly integrate multiple runtimes, enabling synchronous communication between them. This is accomplished through a multi-virtual machine architecture that encapsulates various runtimes, managing transaction execution within a unified framework.

1. Introduction

The prevailing method for connecting different virtual machines and achieving interoperability is to operate them independently as standalone blockchain systems, linked through a communication layer—typically a bridge protocol. Examples of such solutions include Polkadot, Avalanche, and IBC. This approach allows each virtual machine to retain its own networking layer and storage optimizations. The security and stability of each blockchain depend primarily on its own integrity and, in some cases, on the reliability of the communication layer.



However, this method has significant drawbacks. By necessitating that all communications between blockchains are executed asynchronously, it introduces several issues:

- Complex rollback mechanics when cross-chain communication is involved.
- Increased cross-chain vulnerabilities.
- A more complicated user experience.
- Fragmented liquidity.

These trade-offs compromise the efficiency and security of the system, making seamless interoperability more challenging to achieve.

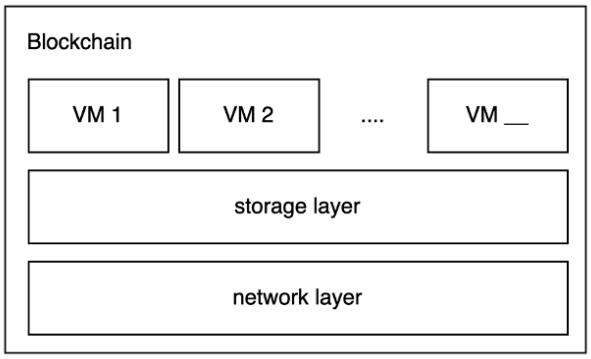
Definitions:

- **Virtual Machine (VM):** A program that executes specific opcode operations within a given context.
- **Network Layer:** The component of a distributed system that manages peer-to-peer communications.
- **State Transition Function (STF):** The mechanism responsible for applying transactions to the blockchain's state.
- **Compute Units (CU):** A metric that quantifies the computational complexity of various opcodes.

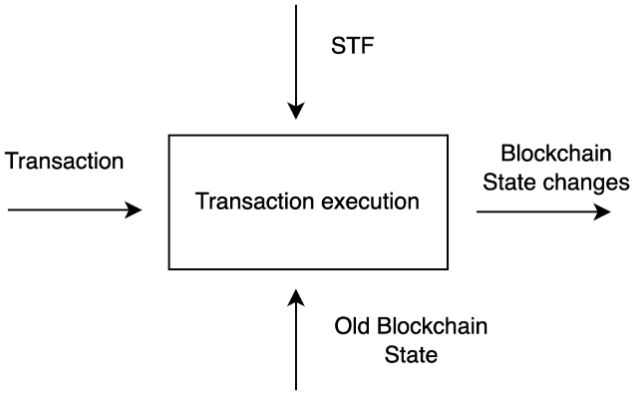
2. Multi-VM

2.1. VM Aggregation

We propose a system where multiple virtual machines operate on the same network layer and shared storage. By aggregating VMs in this manner, we aim to achieve seamless interoperability within a unified framework.



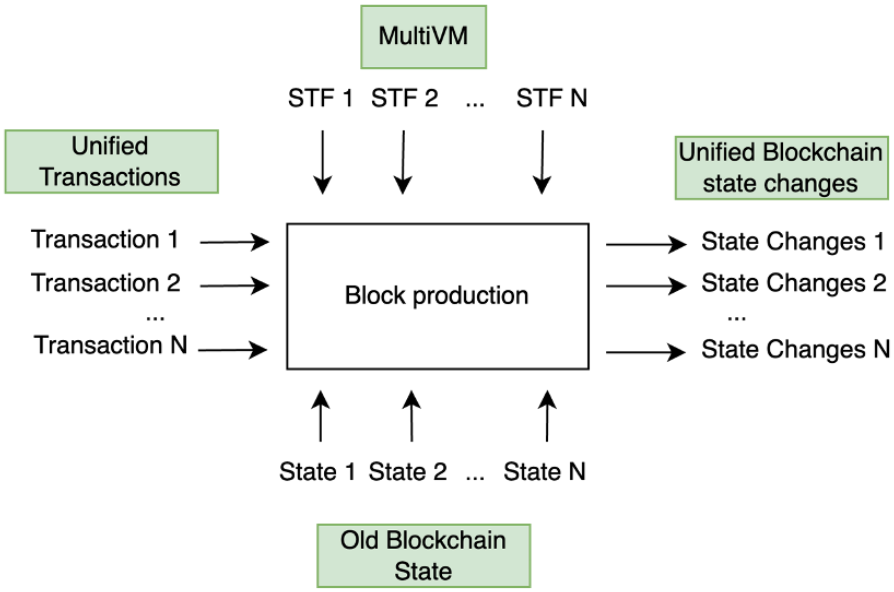
In a decentralized system, every virtual machine fundamentally relies on a state transition function. The following outlines the high-level framework:



The diagram illustrates how the state transition function controls the creation of state changes through transactions and the blockchain state. Typically, blockchains either employ a single type of state transition function by selecting a specific virtual machine or use multiple functions that share the same storage structure, as seen with Arbitrum EVM and Arbitrum Stylus.

2.2. MultiVM Transaction Execution

Since a virtual machine generates state transitions from transactions, multiple virtual machines can operate concurrently on the same logically separated storage. This setup enables a system where the appropriate state transition function can be independently selected for each transaction. As a result, transactions associated with different state transition functions (STFs) can be processed together within a single state transition.



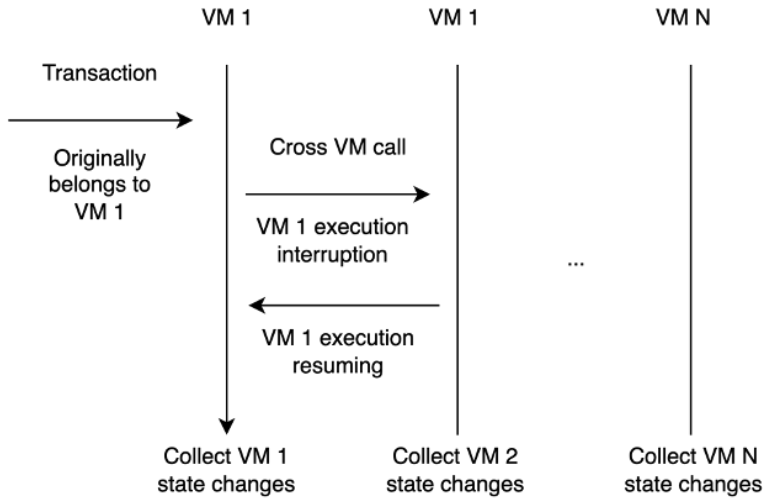
This approach allows any valid state transition function to be integrated into the block production mechanism without interfering with the logic of other STFs. The system still operates as a single STF that encompasses all supported functions, applying unified transactions to a unified state.

2.3. Cross-VM Calls

To achieve seamless interoperability between virtual machines, we introduce Cross-VM calls. These serve as a messaging protocol between runtimes, allowing payloads to be bridged across different virtual machines. By utilizing each virtual machine's native communication methods to perform actions on the destination side, developers can leverage native tools within each virtual machine to build interoperable applications.

Technical Overview

Any transaction can execute a Cross-VM call during its runtime. Each virtual machine independently records its state changes. At the conclusion of the transaction execution, these changes are merged.



When a call is initiated, the execution of the original virtual machine pauses until the call completes. After the call returns, execution resumes with the updated state. Recursive calls to the same virtual machine are permitted, allowing for reentrant calls if the virtual machine itself supports them.

Cross-VM Compute Unit Estimation

When one virtual machine initiates the execution of another, the compute units (CUs) of the transaction are carried forward. A Cross-VM call introduces an additional fixed charge to set up the call context.

While this extra fee eliminates stack limitations for cross-VM calls, the overall execution of the transaction is still bound by the block's maximum compute unit limits.

Different virtual machines have their own equivalents of compute units—for instance, gas in the EVM. When execution shifts to another VM, we use a specific conversion ratio to translate Cytonic's compute units into the native units of the target VM.

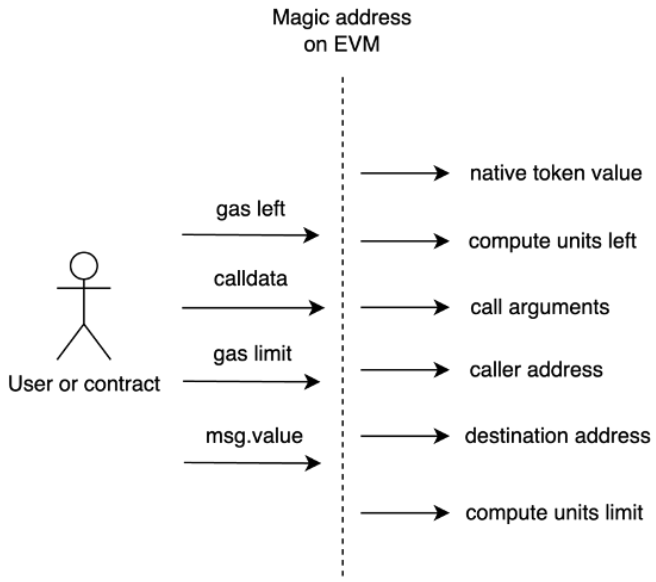
Although the costs associated with VM opcodes may change over time, this does not lead to compatibility issues. The opcode costs are designed to evolve, ensuring ongoing interoperability.

Native Token Transactions

The native token is stored across all supported virtual machine storages. Users can transfer native token balances between different VMs using Cross-VM calls, a feature inherently supported by the system.

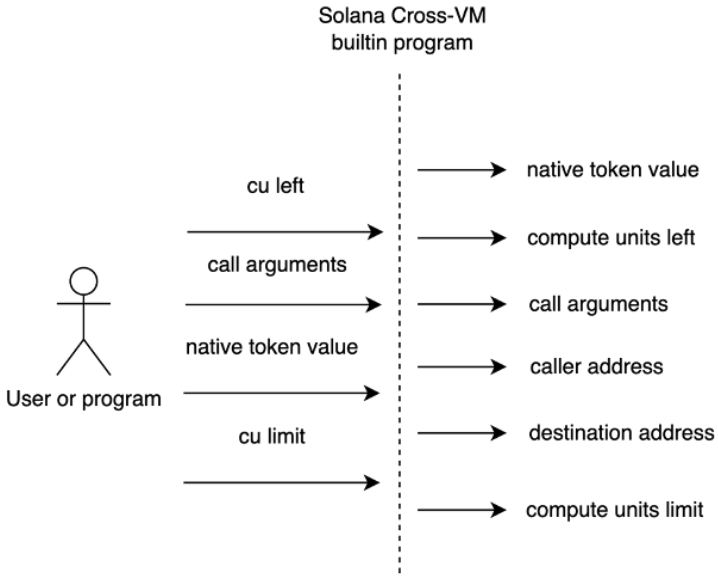
Typically, a Cross-VM call transfers both value and payload simultaneously. However, a Cross-VM call can also have an empty payload, indicating that the call is solely for value transfer. This type of call incurs a lower compute unit charge, as it can be processed more efficiently.

EVM Interface Example
In the EVM, a Cross-VM call is initiated by invoking a call opcode to a specific address. This address isn't owned by any user and doesn't contain a deployed contract; instead, it's a special instruction to the EVM runtime to forward the call to another virtual machine.



The call preserves the entire context, using the EVM's calldata parameter as the arguments for the cross-VM call along with the destination address. Native token values are propagated by including the native token within the EVM call.

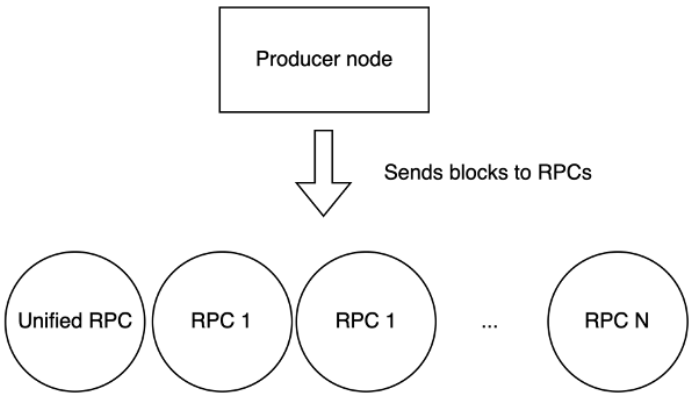
Solana Interface Example
In Solana, cross-VM calls are implemented using a new built-in program. When an account invokes this program, it initiates a Cross-VM call to the target virtual machine. This approach mirrors the EVM interface, where the built-in program acts as a special address.



3. Node Structure

This section describes the part of the Cytonic blockchain architecture that supports multiple RPC standards.

Cytonic's node architecture provides solutions for RPC optimizations tailored to user requirements.
The node structure comprises two main components: the producer node and the RPC nodes.



The producer nodes participate in consensus rounds and perform the following functions:

- Propagate new transactions to the network.
- Validate new blocks.
- Notify RPC nodes of newly validated blocks.

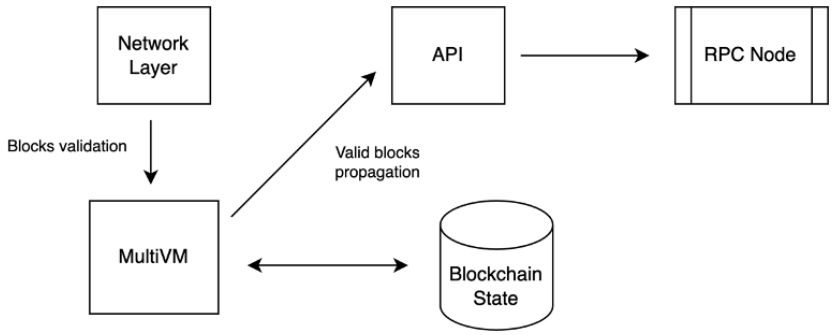
The RPC nodes act as clients to the producer nodes, receiving all network updates and

aggregating blockchain data to ensure compatibility with various virtual machine RPC standards. Their main functions include:

- Enabling compatibility with existing blockchain RPC APIs.
- Simulating transactions and measuring their execution costs.
- Sending transactions by propagating them to any producer node.

3.1. Producer Nodes Structure

The producer nodes communicate directly with the network layer to receive new blocks. They maintain optimized versions of the blockchain state for transaction validation, enabling efficient verification of transactions.

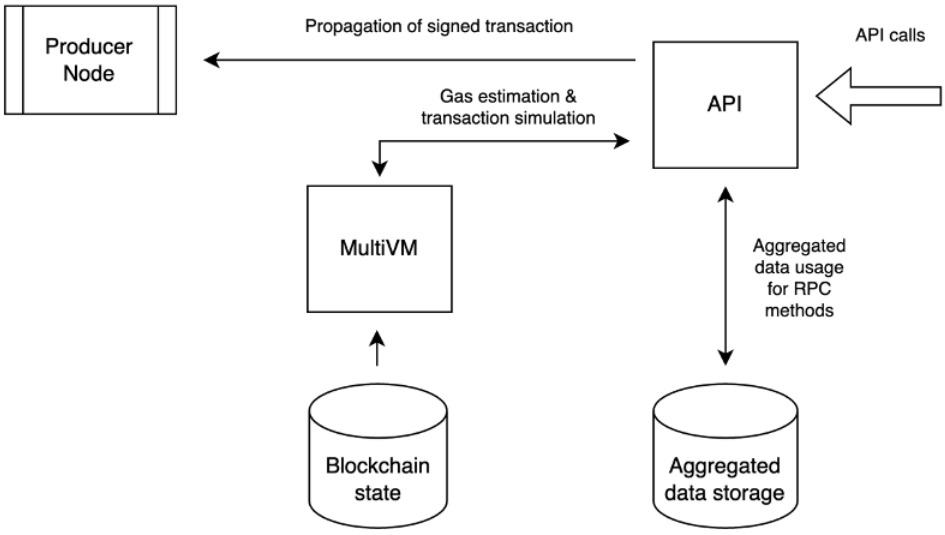


Once a block is finalized across the network, the producer nodes notify all subscribers about the newly created block.

3.2. RPC Nodes Structure

There are two major types of RPC nodes in the network:

- **Unified RPC Nodes:** These provide the most informative and user-friendly way to explore the blockchain.
- **Compatible RPC Nodes:** These focus on ensuring compatibility with existing blockchain API standards.



To operate efficiently, RPC nodes can maintain their own copy of the blockchain state or utilize the producers' state through an API. They also include their own storage for specific aggregated block data.

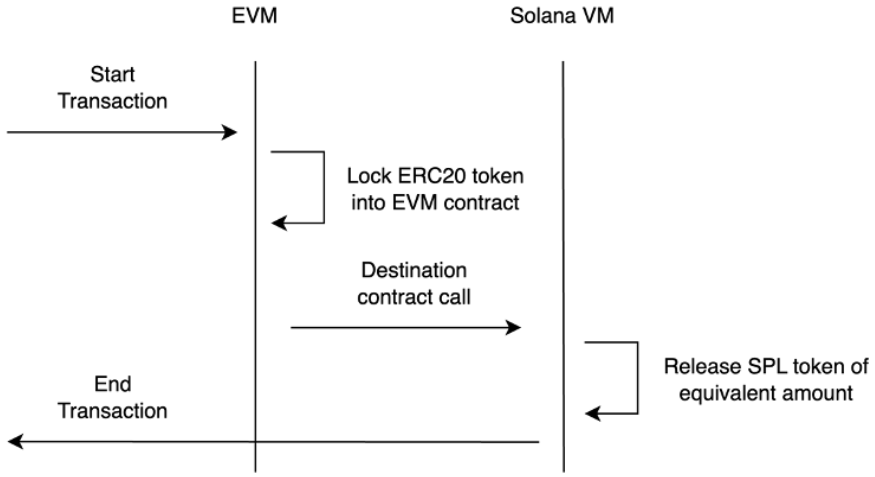
4. Use Cases

This section illustrates future use cases of Multi-VM and its Cross-VM call mechanics in the DeFi space. While these examples primarily utilize the mechanics described earlier, they showcase new approaches for building decentralized financial products.

4.1 Cross-VM Asset Transfer

Cross-VM asset transfers for non-native tokens can be implemented using smart contracts that interact via Cross-VM calls.
This process involves locking the asset on the originating VM and sending information about the locked amount to the destination VM. On the destination VM, an asset of a different standard is then released.

Example: Converting an ERC20 Token from EVM to an SPL Token on the Solana VM



Initiate Transfer on EVM:

- The user calls a predefined contract on the EVM, specifying the amount of tokens to be transferred.
- They provide the destination address on the Solana VM as part of the call payload.

Cross-VM Call to Solana VM:

- The EVM contract performs a cross-VM call to a corresponding contract on the Solana VM.
- It propagates information about the destination address and the locked value.

Issue Tokens on Solana VM:

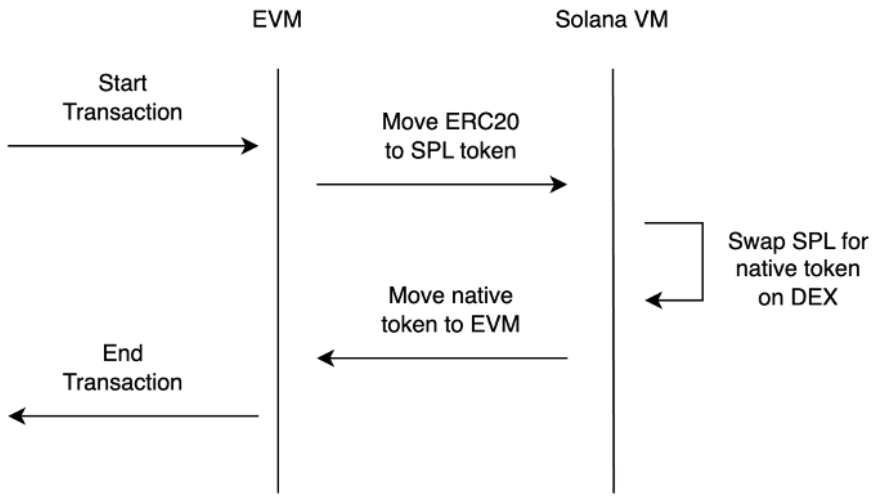
- The Solana VM contract receives the information.
- It issues the required amount of SPL tokens to the specified address.

This approach enables seamless asset transfers across different virtual machines, enhancing interoperability and expanding the possibilities within decentralized finance.

4.2. Cross-VM Automated Market Maker

This example demonstrates how Cross-VM calls enable the infrastructure of one blockchain to be utilized on another. Specifically, it illustrates the process of swapping an ERC20 token for a native token on a Solana VM decentralized exchange (DEX) and then storing the resulting tokens in an EVM account.

This scenario combines the previously described non-native asset transfers with the ability to invoke decentralized applications (dApps) after the transfer.



Initiate Transfer and Swap Request:

- The user transfers an ERC20 token to an SPL token on the Solana VM.
- The token is sent to the DEX address, including a swap and backward transfer call as part of the transaction.

Execution on Solana VM:

- The Solana VM receives the cross-VM asset transfer.
- It swaps the SPL token for a native token on the DEX.
- The native token is transferred to the cross-VM asset transferring contract.

Backward Transfer Execution:

- The cross-VM asset transferring contract performs a backward transfer, which is a cross-VM call with no payload.

Receipt of Native Token:

- The native token is received by the user's EVM address.

This process showcases how Cross-VM calls can facilitate complex interactions between different virtual machines, allowing users to leverage DeFi functionalities seamlessly across different blockchain architectures.

5 Summary

We have introduced Cytonic, a blockchain architecture that integrates multiple virtual machines (VMs) operating on a shared network layer and storage.

By enabling Cross-VM call mechanics, we achieve seamless interoperability between different VMs. This approach enhances transaction validation efficiency and expands the capabilities for decentralized applications.

Cytonic employs producer nodes that participate in consensus and validate new blocks, and RPC nodes that aggregate blockchain data to support various RPC standards. Together, these components improve the network's usability and functionality.

In conclusion, Cytonic provides a unified framework that facilitates high interoperability among diverse VMs, optimizing transaction processes and ensuring compatibility with existing blockchain standards.