Vishwas

by Anika Sharma

Submission date: 29-Nov-2021 11:41PM (UTC+0530)
Submission ID: 1678376456
File name: CloudAssignment.docx (791.88K)
Word count: 1052
Character count: 5166

Submitted by: Arvinder Singh

Roll No: 18124004

Blockchain in integrated Cloud-Fog environment

Blockchain when used in integration with Fog-computing can provide data security and enhances the availability of service. This setup provides us many features like interoperability, encryption of data and compliance to SLA's. Blockchain overcomes most of the research issues of the cloud with its characteristics.

1) Internal Communication Lack of internal communication in public cloud restricts many use cases. This problem can be solved by integrating Blockchain technology, as internal communication is possible in Blockchain. Nodes within a network share data to maintain individual copy of transactions. With Blockchain integration, different clouds are seen as nodes. Hence, provide transparency into the network.

1

2) Encryption and Integrity: One issues is before the data is stored in cloud, it is usually decrypted and might compromise data integrity. Blockchain encrypts data and generates hash code based cryptographic algorithms, and also a hash key is generated for each block. The block discovery consensus mechanisms of Blockchain helps maintain data integrity. The characteristic of blockchain that every full node has its own copy of transaction history maximize data availability and minimizes the downtime.

2

3) SLA's: Service level agreements. These agreements favourable to the service provider or customer without equal justice. The concept of Smart contact in blockchain can provide a solution to this issue, by self-executing a piece of code on all nodes on network, when triggered by a specific condition.

4) Data Management: Blockchain integration provides data storage in a structured manner, which otherwise was stored in unstructured manner. At the same time, it ensures anonymity of user to prevent the access to user's information by third party agent.

Model of Blockchain integration with fog-computing:

- Components of architecture are shown in fig.
- End user (apparently a thin client) can access and interacts with the server via application layer.
- Each transaction request made through the application layer, is stored by creating a block for every transaction.
- 1
- After the validating nodes verified the transaction, based on consensus, the block is added into network.
- The cloud storage used to store this data is indeed blockchain protected cloud storage. Thus, providing transparency and improved services along with data protection.

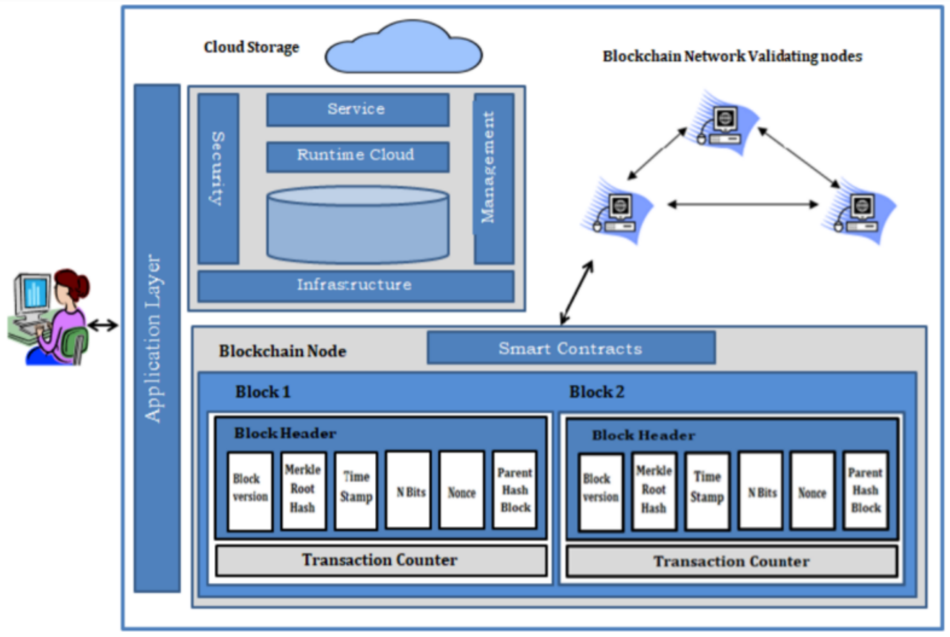

Fig 1. Architecture of Blockchain integrated with Fog-Computing

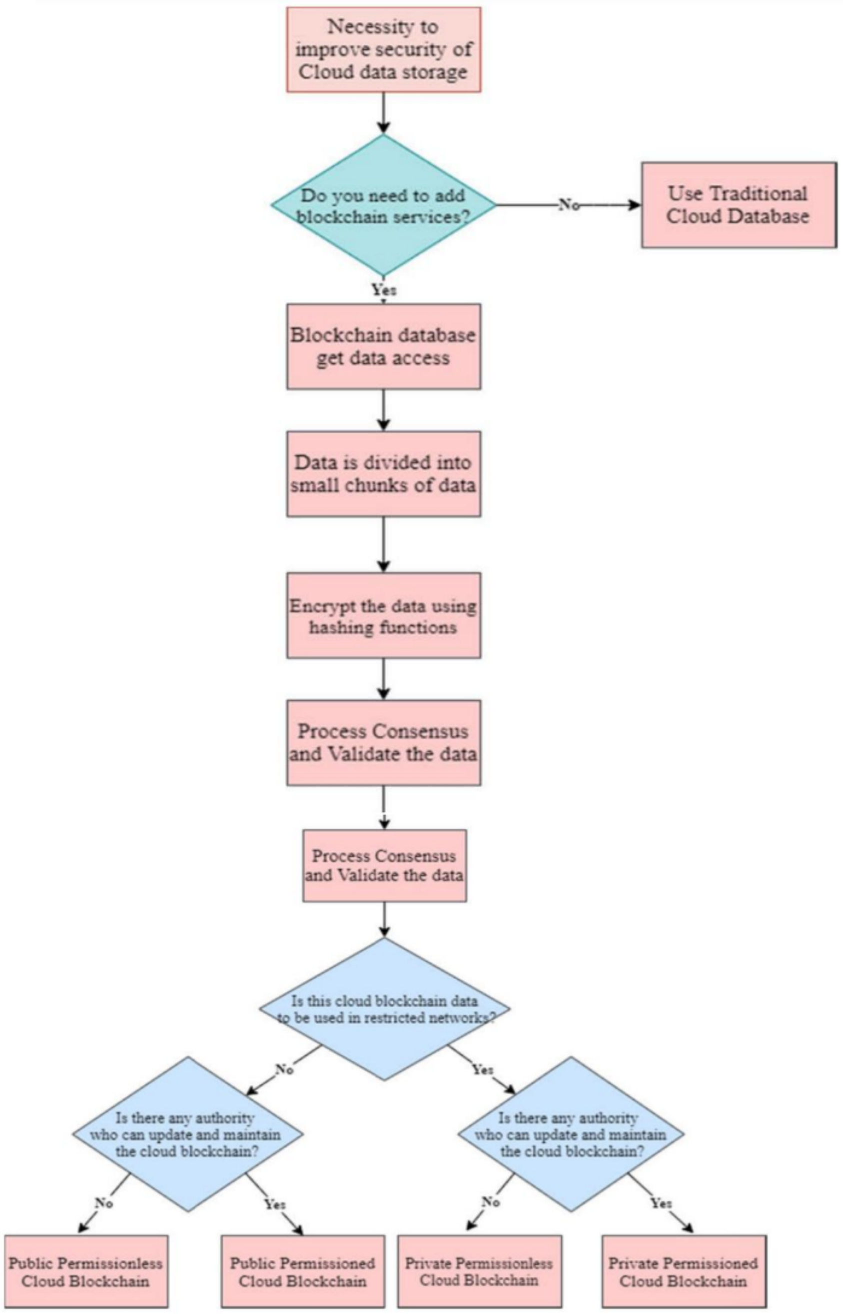
Necessity to improve security of Cloud data storage
Do you need to add blockchain services?
No
Use Traditional Cloud Database
Yes
Blockchain database get data access
Data is divided into small chunks of data
Encrypt the data using hashing functions
Process Consensus and Validate the data
Process Consensus and Validate the data
Is this cloud blockchain data to be used in restricted networks?
No
Yes
Is there any authority who can update and maintain the cloud blockchain?
Is there any authority who can update and maintain the cloud blockchain?
No
Yes
No
Yes
Public Permissionless Cloud Blockchain
Public Permissioned Cloud Blockchain
Private Permissionless Cloud Blockchain
Private Permissioned Cloud Blockchain

Fig 2. Flowchart representing flow of data

Transaction validity:

A node does not trust the information it receives (no room for trust in Blockchain), so it performs a few checks using its own validation protocol. The latter can be seen as a function:

$$\begin{array}{rccc} PrV & : & \mathbb{N} \times \mathbb{E} & \rightarrow & \{True, False\} \\ & & (n, Tr) & \mapsto & PrV(n, Tr) \end{array}$$

where N is the set of all the nodes, and E represents transactions set.

For instance, say, n_e computes the value PrV(n_e, Tr_X). If it is evaluated to False, n_e rejects it otherwise then Tr_X is considered to be a valid transaction by $n_{e.}$ and is broadcasted to the nearby nodes. Then, Tr_x eventually reaches n_c (a complete node) after it enters in network. If n_c finds Tr_x valid, it adds it to its own list of transactions, which is local to n_c , i.e.

$$L_{loc}^{n_c}.append(Tr_X)$$

Eventually a new block (representing a set of valid transactions) is created a out of a subset of transaction in local list of n_c :

$$B_{n_c} = \left(Tr^1, \ldots, Tr^N\right)$$

If $B^{nc}{}_{last}$ denotes the last block in the local chain of n_c, according to protocol, n_c starts its attempt to solve the proof-of-work for ($B^{nc}{}_{last}$, B_{nc}).

Many complete nodes are doing this parallelly i.e., competing among each other. After given time frame, or when sufficient valid transactions are there in local list of a node, the node generates new block and starts solving for POW. After success, presumed winner broadcasts newly created block along with block header. In other words, for a block $B_{new,}$ the nodes claim to have solved POW and others nodes performs their own checks after receiving it. If any of Transaction in B_{new} is found to be invalid, whole block is rejected. And therefore it is checked POW performed by n_c is checked, which is a single hash function computation along with a check on Merkle root of B_{new}.

A complete node n_a , must decide whether to add B_{new} to its local blockchain. If the block B_{new} is added to the local version of its blockchain, then hash of the previous block (B_{prev}) in transmitted header Head (B_{new}) and the hash of the last block in the local blockchain are equal. Since n_a has accepted the B_{new} , it removes the set of transactions present in this block from its local list and its miner is rewareded as per protocol.

Here, problem of conflict of version might arise. Say at time $t_{0,}$ all the nodes agree to same version of blockchain, say $BK_{0.}$ And let B_1 and B_2 be two blocks broadcasted by two different nodes at some time t_1 ($t_1 > t_0$). Since there might be some time delay for blocks to reach some nodes, a condition may arise when some nodes receive B_1 before B_2 or in the other way, leading to condition of fork. Formally speaking, some nodes will have their local version of blockchain as

$$BK_0 + B_1$$

While for some nodes, their local version of blockchain will look like

$$BK_0 + B_2$$

The rule of consensus, which forms the security of blockchain, says that a full node will always keep the version of chain with largest work i.e., the longest chain. Say at some time t_3 ($t_3 > t_2$), one of the nodes with the version of local blockchain $BK_0 + B_1$ emits a Block B_3 into the network. Means, the local blockchain for this node will look like $BK_0 + B_1 + B_{3.}$ When the broadcasted block B_3 reaches a node (say n_f) with local blockchain $BK_0 + B_{2,}$ there will be a problem. Since, B_3 needs B_1 as previous block, it can't be added to the local chain of n_f. This is a condition of a dilemma for n_f. Since dilemma is for the blocks after BK_0, there are two options: $BK_0 + B_1 + B_3$ and $BK_0 + B_2$. By protocol, longest chain will be kept i.e., $BK_0 + B_1 + B_3$, and transaction of block B_2 are again taken into consideration for new block. As the depth of chain goes on increasing, the confidence for the deeper nodes keeps on increasing in the whole blockchain network. Ultimately all nodes in network achieves the consensus.

Vishwas

ORIGINALITY REPORT

6%	0%	6%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Ch.V.N.U. Bharathi Murthy, M. Lawanya Shri, Seifedine Kadry, Sangsoon Lim. "BLOCKCHAIN BASED CLOUD COMPUTING: ARCHITECTURE AND RESEARCH CHALLENGES", IEEE Access, 2020 Publication	5%
2	Ch. V. N. U. Bharathi Murthy, M. Lawanya Shri, Seifedine Kadry, Sangsoon Lim. "Blockchain Based Cloud Computing: Architecture and Research Challenges", IEEE Access, 2020 Publication	1%

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		