Application Engine using Blockchain Technology

A. Jasmine Gilda, Department of Computer Science and Engineering, R.M.K. Engineering College, Chennai, India. Email: ajg.cse@rmkec.ac.in
A. JebaSheela*, Department of Computer Science and Engineering, DMI College of Engineering, Chennai, India. Email: jebasheela.soft@gmail.com
Dr.Sandra Johnson, Department of Computer Science and Engineering, R.M.K. Engineering College, Chennai, India. Email: sjn.cse@rmkec.ac.in
Dr. T. Sethukarasi, Department of Computer Science and Engineering, R.M.K. Engineering College, Chennai, India. Email: tsk.cse@rmkec.ac.in

Article Info
Volume 82
Page Number: 8840 - 8844
Publication Issue:
January-February 2020

Article History
Article Received: *5 April 2019*
Revised: *18 Jun 2019*
Accepted: *24 October 2019*
Publication: *08 February 2020*

Abstract:
Internet connections have become prevalent and data has become cheaper due to competing Internet Service Providers and various other factors. This creates an opportunity for using the idle time of computers whose performance has increased exponentially over the years to create a network through which their computing power can be better utilized. By including a huge number of computers data sharing can be made easy which in turn paves the way for a open and reliable Internet. In the core of the network a Blockchain is used to overcome the challenges on security. Every considerable action performed on the network is recorded on the Blockchain to provide accountability. Censorship and privacy woes are tackled using the combination of peer-to-peer networking, de-centralization and immutability. Improvements on the Rapid Application Deployment model are introduced.

Keywords: *distributed computing, peer-to-peer networking, blockchain, security*

I. INTRODUCTION

Cloud computing has revolutionized how web applications are deployed and served [3]. By providing features as services the need for huge private infrastructure has been eliminated. Instead based on the needs of an application a suitable service can be selected and then used without needing the complete infrastructure. Multiple instances are run using virtualization technologies in a single or interconnected multiple powerful cloud servers [4]. Various instance management systems are used to configure these systems and to maintain them. We use this model of providing features as services to create an application engine which provides the service of a platform. On this platform we deploy user applications which can be configured and published by the users.

Instead of using a single server model, a distributed computing model is used to serve the applications published on this network. Distributed computing is implemented using a peer-to-peer networking model. To increase reliability, application code and resources are replicated to multiple computers connected to the network. Load balancing and sharing approaches are enabled to serve application from the an optimal point.

II. DECENTRALISATION

Current cloud providers are large but in small numbers. This leads to monopolisation of cloud resources and data [5]. In unfortunate circumstances this may also lead to censorship and other ways of control. Decentralisation can help tackle this problem by distributing resources and spreading control from concentrated points. In turn this also provides with reliability and redundancy. Decentralisation brings certain challenges along with its advantages such as scalability. But the advantages over a worth case monopolisation is much better than the technological challenges that could be faced. It also allows for better utilization of resources spread across a wide variety and range of systems.

III. Rapid Application Deployment

The aim of the network is to enable developers create applications which can be served to multiple users without spending much resources on the infrastructure. Instead of spending time on setting up the necessary infrastructure, users can implement continuous development and integration on their applications and publish their changes to the network where the infrastructure is automatically taken care of. By providing flexible options for configuration on the platform the number of systems and locations to deploy the power of the network can be leveraged. Thus by how applications are continuously improved with continuous integration and development, applications can be served better using Rapid Application Deployment.

IV. Application Engine

To publish applications on the network, the user needs to use the Application Engine which follows a general application structure for its applications. Each application is contained in an application folder which comprises of certain mandatory files as explained in the following Application Folder section. By making use of the information present in the mandatory files the applications are deployed and managed on the network.

The Application Engine consists of multiple components of which each component performs a specific function of its own. The components and their function are explained in the following sections.

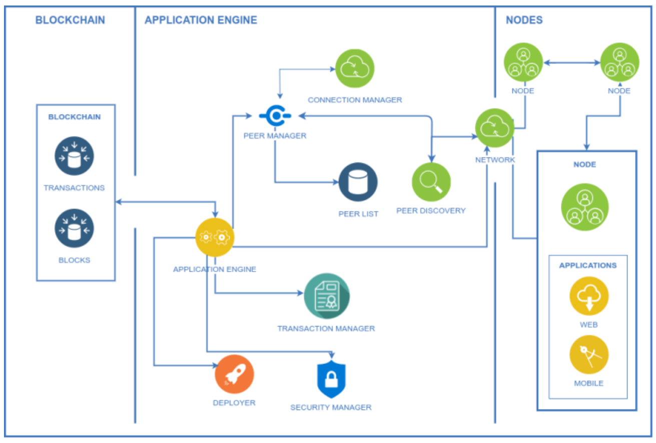

Figure-1. Architecture of the network

A. *Application Registry*

Applications deployed or published in the network are automatically aggregated and presented to the user on the application registry. The application registry provides the user with a web application view through which the user can view details regarding the application. Transactions which occured on an application can be viewed along with its general configuration. It also provides options to performing a transaction on an application.

B. *Replicator*

Since the network requires files to be replicated from one node to another for data and application sharing, a separate module handles the synchronisation. It maintains the folder structure and transfers files which have to be deployed on the network from one node to another. Replication only happens when an user approves an applications

C. *Security Manager*

The main security concerns regarding applications published in the network are tampering of application data and code and malicious code execution. Security manager ensures of safety of the application by performing a variety of operations like hashing and verifying content integrity.

1. Hashing

Hashes of all files are periodically recorded by the security manager. Hashes of the files are recorded in the app info using the SHA-256 algorithm.

2. Integrity verification

Using hashes recorded by the Application Engine in the app_info file, changes made to the files by users other than the application owner are tracked.

D. *Transaction Manager*

Every transaction is performed by the transaction manager. Transactions are tried to be kept atomic as possible. It is also responsible for writing the records into the files.

E. Deployer

Applications are deployed when the user finishes making changes to the application and then initiates the deploy transaction. After the initial transaction, deployments are automatically done based on the changes and the information on the application information file.

F. Peer Manager

List of the peers to which connections can be made to retrieve files, changes to the shared Blockchain and other operations are maintained by the Peer Manager. It can initiate a connection to another peer by using the connection manager.

G. Connection Manager

Connections are only made to other peers are made only when an action such as replication or deployment is performed. This helps to keep unwanted connections and load on the network to be minimum as possible.

H. Load Balancer

Applications are served through a load balancer running on any of the alive nodes. The load balancer component is present in nodes which include the option and are selected in a round robin robin fashion. Node hosting the content for an application is find by resolving the unique addresses through a series of nodes.

V. APPLICATION FOLDER

Each application is contained in a folder which consists of the application code and the files mandatory to create an application on the network.

A. Folder structure

Each application contains mandatory files as described below. The entry point of the application is defined in the configuration file of the application. Application code is contained in the app folder.

1. transactions.json
All transactions performed in an application is recorded in the transactions.json file.

2. app_config.json
Configuration specific to an application is stored in the app_config file. The hash of the config file is stored in the app_info file.

3. app_info.json
This file contains all the files present in the application directory,Blockchain transactions require the hash of this file to create a transaction.

VI. APPLICATION BACKEND

Currently, web applications are supported on the platform. This is achieved by providing an application layer on the platform which has the components for running a web server and the necessary modules. Since the web applications have to be deployed to multiple user machines, it is necessary for the web component to work with different versions of the Application Engine. Backward compatibility and separation of the web backend module from the Application Engine have to be maintained. Multiple layer separation are currently proposed in the application backend. This enables multiple frameworks and languages to be supported using the same backend. To protect user information and to avoid vulnerabilities introduced by the user in the application to compromise the security of the system the application is running on, containerization methodologies are used. This provides compatibility and isolation in addition to security.

VII. TRANSACTIONS

To maintain accountability on the network, actions which are of importance are performed by initiating a transaction and then proceeding with the action. Most transactions are recorded locally on a blockchain-like structure and periodically synced with the network.

A. Application Transaction

Every action performed on an application is recorded as a transaction on the transactions.json file in the application folder. Transactions written on the transactions.json file is different to the high level transaction written on the file.

Each transaction consists of a particular type along with a timestamp and other necessary parameters. The following transaction types are followed.

1. Local
This is the initial transaction recorded when the user creates an application. Changes made to an application before publishing are recorded as a local change.

2. Audit
When an user wants to inspect an application or clone the application to deploy it on their node, the audit transaction is performed. Replication of files start after this transaction.

3. Approve
When an user verifies an application it is termed as an approve transaction. To deploy an application on the network and to use other nodes as points of deployment, it needs to be approved by a minimum number of users.

4. Deny
Applications which are termed as malicious can be denied by initiating a deny transaction. If an application receives deny transactions greater than the current threshold it is removed from the application registry.

5. Deploy
When an local application is deployed into the network, a deploy transaction is performed.

B. Blockchain Transaction

Blockchain transactions are only written to the Blockchain which is synced across all the nodes connected in the network. This helps to keep the size of the Blockchain compact.

1. Approve
This is the initial transaction recorded when the user creates an application. Changes made to an application before publishing are recorded as a local change.

2. Deny
This is the initial transaction recorded when the user creates an application. Changes made to an application before publishing are recorded as a local change.

VIII. BLOCKCHAIN

All transactions which contain crucial information such as the hashes of the app_info file, tokens spent and timestamps are written to the Blockchain. Sufficient care is taken to keep the size of the Blockchain compact. A tree-like data structure is used to enable fast retrieval of transactions from the Blockchain. The Blockchain is a data structure which consists of a number of blocks. Each block consists the hash of the previous block present in the chain of blocks. If the hash of a block is changed, it will no longer be equal to the hash in the next block. The last block of the chain will be not written to the Blockchain but kept in the memory of the user machines.

A. Blocks

Each block consists of a number of transactions each of which is a transaction specifying the action performed by an user on an application. Once a block is written to the Blockchain it cannot be changed. If the Blockchain is tampered with, then other nodes in the network will stop accepting connections or sending connection requests to the node which holds the tampered Blockchain.

B. Transactions

Each transaction contained in a block consists of multiple parameters. Transactions are arranged as trees in the blocks and headers are stripped and stored in nodes which don't want the full Blockchain.

IX. TOKENS

When the network grows, incentives should be provided for hosting applications of other users. This is achieved by tokens which can regarded as a form of currency and can be exchanged for other monetary forms. To publish an application, certain number of tokens have to be spent which keeps the network from being flooded with rogue applications. It is necessary for applications to be approved by a mechanism such that malicious applications don't get an higher approval count. Token relations transactions are to be implemented in future and will be written to the Blockchain.

X. Further Work

By providing applications which interacts with the whole network to obtain computing resources, tasks requiring high computational complexity may be performed. A generalised interface where the problem, algorithm and input can be provided which in turn is executed on the whole network and results as output is derived can be also be integrated.

XI. References

[1] Communications Today: Indian Telecom Market is Going Through A Phase of Disruption October 24, 2017
[2] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system.
[3] CIO: 6 Amazing Advances in Cloud Technology April 14, 2014.
[4] RedSwitches: The Different Types of Virtualization in Cloud Computing – Explained, June 1st, 2017.