Smart Contract Monitoring using Cloud Driven Technology

1st Dawid Pionk   
*South East Technological University*Carlow, Republic Of Ireland  
Dawidpionk01@gmail.com

*Abstract*—This paper explores the possible uses for cloud computing with the blockchains and vice versa. It aims at trying to connect decentralized blockchain with centralized off-chain databases through the use of smart contracts. This project was conducted using several technologies. Aws for cloud storage. Node.js using several libraries for event listening for changes on the blockchain. Solidity smart contracts previously deployed on the blockchain. Infuria was used to connect to Ethereum nodes. Foundry was used for testing and casting onto the blockchain. Then the system was evaluated to see how it performed against set out goals. Then the systems limitations were examined. The findings in this paper suggest that this type of hybrid system would be right for some system.

Keywords—Blockchain, Cloud Computing, Smart Contracts Monitoring, Game Transactions

# Introduction

In this paper, we will present a review of how cloud computing can be interacted with on the blockchain. More specifically we explored the area of Product as a service. This brought several limitations such as security etc into play. We think this is a valuable field of research since it could offer more security and cheaper ways of storing data. Some other interesting parts include decentralized access to data and some real word use case research. With this paper we contribute to the field by researching hybrid systems and developing a prototype for a cloud to blockchain application.

# Literature Review

In this section I give an overview of the research I done for this project, what implementation of integrated blockchain and cloud services I found. I also discuss other peoples research and how it aided me over the course of this project.

Cloud computing interaction with blockchain is not a common interaction of services. I think it could be useful since it offers, an easier storage of data and way for people to interact with the blockchain if they don’t understand it. It offers several benefits like offering storage of data, databases, easier monitoring of the blockchain and so on. You can also host nodes themselves on cloud services

An interesting case study for a cloud system that integrates cloud computing is the IBM food trust. Under this system blockchain was used to enhance transparency and efficiency. It allowed for a secure way of tracking transactions and food products to avoid waste, fraud and to track origin of food products that spread diseases. This system also allows for the creation for a standard ledger that could be used by all members of a particular supply chain. How it works is by only giving permission to trusted participants to enter data, like farmers or shopkeepers. Then those shopkeepers instead of entering their data into a paper or excel ledger would enter it into the food trusts distributed ledger on the blockchain, allowing everyone easy access to view. [1]

In a paper called about blockchain based cloud computing by Bharathi Murthy et al. analyses how integrating cloud computing with blockchain technologies can be used to mitigate certain key limitations in the field of cloud computing by surveying previous papers that have been written on this subject. It looks at how things like privacy, data security and management could be improved through the blockchain. It explains each of the technologies in the great detail and proposes briefly a hybrid system using both technologies. This paper proved useful towards my proposed system cause it showed how I can combine the 2 systems. [2]

In a survey done by Jinglin et al. called “Integrated Blockchain and Cloud Computing Systems: A Systematic Survey, Solutions, and Challenges” they explore the combination of blockchain and cloud computing systems. It is focused on how hybrid system works the different combinations and improvements to security. It also explores other subjects such as consensus protocols, recent industry trends and different models. It also explores different gaps in research and some challenges such as performance issues, the energy costs and compliance with different regulations. This survey helped me explore different models of blockchain and cloud computing models that I could implement into my own research, as well as some interesting implications of it. [3]

An interesting article by the Swiss cyber institute talks about how blockchain technologies can enhance security in cloud environments. This article talks about how blockchains security systems such as immutable blocks and consensus protocols could be used to provide additional security. It also talks about how its decentralized nature allows for greater resistance against failure or attacks. What I gained from this article was the benefits of security blockchain systems provide [5].

I also looked into other poterntial applications of blockchains to give me an idea of how to best apply my project. I looked into an article to better understand the benefits of an integration of the technologies could provide for scalability and security. [6]

How I plan to imply the research I done into the project I’m creating is by looking through this and choosing what particular aspects from each paper I want to apply. How I am going to integrate the traditional cloud system and see how I can apply it to my project.

# System Architecture

In this section I give an overview of the technologies I interacted with for this project. I give a high level overview of how the project was set up. I explain each of the individual components that were used to implement the project and its technologies, I also include a diagram to aid in the explanation of this system.

## Technologies

So how cloud computing interacts can be bit complicated but has many applications. In this report I decided to focus how decentralized blockchain could interact with centralized cloud data centers. This process can be complicated since blockchains don’t have a direct way to interact with cloud data centers.

## Components

Below I explain how each component interacts with the overall system:

* Cloud Server: This is where the data can be stored or retrieved from. Examples of these include AWS or Azure
* Blockchain: Here is the system where the underlying Ethereum is.
* Smart Contracts: They are hosted on the blockchain, these are programs where the users can interact with the blockchain or different data.
* Web3 programs: Here the user can use as a point to interact with blockchain from.
* UI: Here is the actual interface the user interacts with
* Lamba function: This listens out for the changes on the blockchain.

## Architecture diagram

# Experiment setup and performance evaluation

After the text edit has been completed, the paper is ready for the template. Duplicate the template file by using the Save As command, and use the naming convention prescribed by your conference for the name of your paper. In this newly created file, highlight all of the contents and import your prepared text file. You are now ready to style your paper; use the scroll down window on the left of the MS Word Formatting toolbar.

## Environmental setup

How this project was setup was through several steps:

1. Created a database on the aws servers where the data be stored
2. Created a table on the said database.
3. Created a smart contract using solidity that deals with creating vaults.
4. Created an project key using infuria so I would be able to interact with the sepolia testnet.
5. Deployed smart contract onto the testnet.
6. Created an event listener using javascript and node.js that listens out for any events emitted by the smart contracts address.
7. Tested it out by creating a vault and seeing if the table was updated.

## Use cases

The primary use case for this specific experiment would be users who want an easy way to view and store data produced on the blockchain. This system shows how a proposed hybrid blockchain to cloud data storage system could work. When a vault is created the data is stored both on the blockchain and on a AWS database. This has benefits such as users having easy access to view data, as well as data stored on servers being easy to audit since the blockchain can’t be altered. This system could be expanded into areas that need to be easy to access and couldn’t be altered such as secure document storage or encrypted medical data.

## Evaluation

We managed to achieve several functionality goals:

1. The smart contract was successfully deployed onto the blockchain.
2. The event listener successfully detects addVault() events.
3. The table on the AWS database was successfully updated after an added vault.
4. The system doesn’t miss any events when it is actively listening.
5. The system takes under 6 seconds to go from emitted event to updated table.

# Conclusion

In this I presented a model for gaming system interacting and taking advantage of both the blockchain and cloud services. It solves issues of things like security and fault resistance in products that could be wiped out by a faulty server or stolen if not secure. The experiment performed showed that the proposed system is feasible. This research also opens the door to other potential areas that this system could be applied to such as aiding supply chain systems and other areas. Overall the findings have the potential to aid future cloud services to be more secure.

##### Acknowledgment

I would like to express my thanks to my supervisor Shei Lei for their guidance through this project as well as the staff at UNUM for their classes that enhanced my understanding of cloud computing,

##### References

1. G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. *(references)*
2. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
3. I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.
4. K. Elissa, “Title of paper if known,” unpublished.
5. R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.
6. Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].
7. M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.

##### aPPENDIX

Code Listing A. GameState Smart Contract

// SPDX-License-Identifier: UNLICENSED

pragma solidity ^0.8.13;

contract GameState {

    struct Item {

        uint256 id;

        string name;

        uint256 price;

        uint256 attack;

        uint256 defense;

        address owner;

        bool forSale;

    }

    event ItemBought(

        address player,

        uint256 quantity

    );

    event ItemCreated(

        uint256 indexed itemId,

        string name,

        uint256 price,

        uint256 attack,

        uint256 defense

    );

    Item[] public itemsList;

    modifier onlyDeveloper() {

        address Developer = 0x1234567890123456789012345678901234567890; // Replace with the actual developer address

        Developer = msg.sender; // This here just for the sake of testing

        require(msg.sender == Developer, "Not the contract owner");

        \_;

    }

    modifier onlyPlayer(uint256 itemId) {

        require(msg.sender == itemsList[itemId].owner, "Not the player");

        \_;

    }

    function createItem(

        string memory name,

        uint256 price,

        uint256 attack,

        uint256 defense

    ) public onlyDeveloper returns (uint256 itemId){

        itemId = itemsList.length;

        Item memory newItem = Item({

            id: itemId,

            name: name,

            price: price,

            attack: attack,

            defense: defense,

            owner: address(0),

            forSale: true

        });

        itemsList.push(newItem);

        emit ItemCreated(itemId, name, price, attack, defense);

    }

    function buyItem(uint256 itemId) public payable {

        require(itemId < itemsList.length, "Item does not exist");

        Item storage item = itemsList[itemId];

        require(msg.value >= item.price, "Not enough Ether sent");

        require(item.owner == address(0), "Item already owned");

        if (msg.value > item.price) {

            payable(msg.sender).transfer(msg.value - item.price);

        }

        // Below section I send the money to the developer in this case I set it as the sender

        // just for the sake of testing

        address developer = msg.sender;

        payable(developer).transfer(item.price);

        item.owner = msg.sender;

        item.forSale = false;

        emit ItemBought(msg.sender, itemId);

    }

    function buyItemFromPlayer(uint256 itemId) public payable {

        require(itemId < itemsList.length, "Item does not exist");

        Item storage item = itemsList[itemId];

        address seller = item.owner;

        require(seller != address(0), "Item not owned by anyone");

        require(item.forSale == true, "Item not for sale");

        require(seller != msg.sender, "Cannot buy your own item");

        require(msg.value >= item.price, "Not enough Ether sent");

        /\* Here the service fee would be deducted

        This section removed for the sake of the experiment

        (bool platformSuccess, ) = payable(developer).call{value: platformFee}("");

        equire(platformSuccess, "Payment to platform failed");

        \*/

        // Refund excess Ether

        if (msg.value > item.price) {

            payable(msg.sender).transfer(msg.value - item.price);

        }

        // Pay the seller

        payable(msg.sender).transfer(item.price);

        // Transfer ownership

        item.owner = msg.sender;

        item.forSale = false;

        emit ItemBought(msg.sender, itemId);

    }

    function getItem(uint256 itemId) public view returns (Item memory) {

        require(itemId < itemsList.length, "Item does not exist");

        return itemsList[itemId];

    }

    function setItemPrice(uint256 itemId, uint256 newPrice) public onlyPlayer(itemId) {

        require(itemId < itemsList.length, "Item does not exist");

        Item storage item = itemsList[itemId];

        item.price = newPrice;

    }

    function setItemForSale(uint256 itemId, bool forSale) public onlyPlayer(itemId) {

        require(itemId < itemsList.length, "Item does not exist");

        Item storage item = itemsList[itemId];

        item.forSale = forSale;

    }

    function getItems() public view returns (Item[] memory) {

        return itemsList;

    }

    function getItemCount() public view returns (uint256) {

        return itemsList.length;

    }

}

Code Listing B. listener.js

const { ethers } = require("ethers");

const AWS = require("aws-sdk");

// AWS Region setup

AWS.config.update({ region: "eu-north-1" });

// DynamoDB client

const dynamoDb = new AWS.DynamoDB.DocumentClient();

// Ethereum provider (Sepolia via Infura)

const provider = new ethers.JsonRpcProvider("https://sepolia.infura.io/v3/12be543cebb045c3b94f5820ef89dc7b");

// Contract address and ABI

const contractAddress = "0x80a712A480292f5EFd3aBb387B6fC56d9B8fdf29";

const abi = [

    "event ItemCreated(uint256 indexed itemId, string name, uint256 price, uint256 attack, uint256 defense)"

];

// Setup contract

const contract = new ethers.Contract(contractAddress, abi, provider);

console.log("Listening for ItemCreated events...");

// Event listener

contract.on("ItemCreated", async (itemId, name, price, attack, defense) => {

    console.log(`ItemCreated event: ID=${itemId}, Name=${name}, Price=${price}, Attack=${attack}, Defense=${defense}`);

    const params = {

        TableName: "itemEvents",

        Item: {

            itemId: itemId.toString(),

            name: name,

            price: price.toString(),

            attack: attack.toString(),

            defense: defense.toString(),

            timestamp: new Date().toISOString()

        }

    };

    try {

        await dynamoDb.put(params).promise();

        console.log("Stored item event in DynamoDB.");

    } catch (err) {

        console.error("Error storing event in DynamoDB:", err);

    }

});