Implementing a Microservices System with Blockchain Smart Contracts

1st Shreyas Sakariya
DA-IICT
Gandhinagar, India
201901075@daiict.ac.in

2nd Ronak Parmar
DA-IICT
Gandhinagar, India
201901102@daiict.ac.in

3rd Ankit Rathod
DA-IICT
Gandhinagar, India
201901108@daiict.ac.in

Mentor : Jayprakash Lalchandani
DA-IICT
Gandhinagar, India
jayprakash_lalchandani@daiict.ac.in

***Abstract*—Blockchain technology and the Microservices Architecture Style (MSA) are two interesting study areas in computer science that attract researchers' interest since they introduce new opportunities for innovative applications. Results demonstrate that a straightforward microservices-based system may be implemented using smart contracts while still preserving the same set of features and outcomes. In this paper, we presented the solution to the defined problem with the help of Blockchain Technology.**

I. Introduction

A methodology to application development known as microservice architecture divides a big program into groups of independently deployable services, many of which are Task-specific. They have several benefits due to their autonomous deployment. They may be created using many programming languages, and they can scale separately. They may be placed on the hardware from other services that satisfies their requirements. On the other hand, a blockchain is simply a decentralized database that is distributed and contains blocks of linked transactions. The symmetric paradigm also applies to smart contracts, which represent executable programs that are clearly specified, normally isolated, and typically accomplish simple activities that have a clear purpose. These tasks may be understood to be services that the contract is providing.

When blockchain and microservices are combined, businesses will be encouraged to establish more partnerships and act as a pipe between partners without the integrity and security of products and services. Blockchain can simplify things by acting as a facilitator for relationships of trust. In our study, we analyse the benefits and drawbacks of the two strategies and show how the newly developing blockchain technology may be easily modified to effectively replace another current software technology and design.

II. Background Reading

A single application is developed using the microservices architectural design as a collection of small services, each of which runs in its own process and communicates using simple tools. In this article [6], we examine a number of microservices-based design features, as well as numerous possible applications in the aerospace sector. the componentization, structure, endpoints, and messaging methods of a microservice-based architecture. the technological implementation of microservices by looking into services communication, containerization, and associated architectural elements.

While cryptocurrencies still garner most of blockchain's attention, smart contracts have emerged as a key use case. Similar to classes, smart contracts may be invoked by client apps running outside of the blockchain. Therefore, it is feasible to create blockchain-oriented software (BOS) that uses smart contracts to execute a portion of the business logic in the blockchain. There isn't yet a design standard for modelling BOS.In this paper [4], we show three complementary modeling approaches based on well-known software engineering models and apply them to a BOS example.

From paper Ethereum Smart Contracts as Blockchain-oriented Microservices [2], we analyze the parallels between the two paradigms and offer a model of software architecture in which microservices are built using Smart Contracts distributed on a blockchain. They also provide an example of the development of an e-commerce site.

From the paper [3]we study the case, analyzing the chronology of the events and the source code of the smart contract library. Also found that the vulnerability of the library was mainly due to a negligent programming activity rather than a problem in the Solidity language.

III. Microservices System with Blockchain Smart Contracts

It is possible to lower risk, boost productivity, and use smart contracts to automate the execution of business logic by designing blockchain applications as microservices. Blockchain smart contracts and microservices have a lot in common. They are both supposed to run independently(on-chain) and connected with the outside (off-chain) via a message-based channel. Considering the complexity of the use of blockchain technology in a variety of applications, the is usually not a simple task to analyze such applications.

IV. Microservices Based Architecture

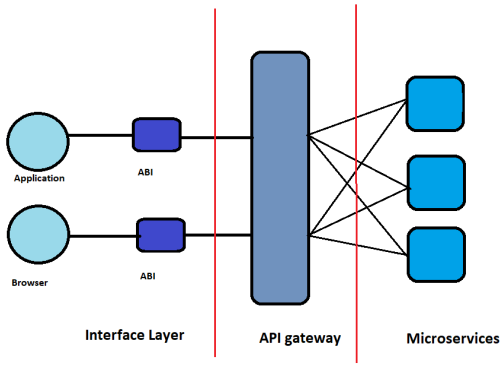

Fig. 1. Microservices Based Architecture

As we can see it contains three layers. The first layer is the Graphical user interface which provides an outside view of architecture to the user. The second layer is API Gateway, which provides a custom interface for each type of device, call different microservices and the return result to the caller back, and does load balancing. The third and most important layer is Microservices, which commonly published the message into the channel bus so that other microservices can subscribe that message to the same channel and read the massage signal in getting the result.

Here, the complete system doesn't need to be pulled down and put back into service. This makes it possible for the software system to operate and undergo updates without any issues. Scaling up gets simpler as a result. Deployment is made simpler and faster with smaller codebases. Designing a module with just that module's functionality in mind is simpler. Each module's specialized functioning is simpler to understand than the others.

By dividing a program into several smaller modules, the communication overhead increases. It could be a difficult and tough process to deploy. During deployment, there would need to be a collaboration between several services.

V. Blockchain Based Architecture

It consists two-layer. The first layer is the same as described above Interface layer, which is between the external applications and the blockchain. It provides the application Binary Interface(ABI). The second layer is composed of a set of smart contracts deployed into the blockchain. Here microservices are implemented by the single atomic smart contacts.

By using blockchain we get decentralized trust, with no single point of failure. Fewer transactions may be processed per second with blockchain technology. It results in poor scalability by delaying the completion of a huge amount of transactions. However, a number of solutions have been put out to address this weakness, but none have been implemented as of yet.

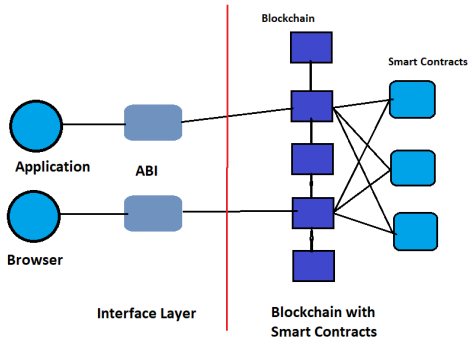

Fig. 2. Blockchain Based Architecture

VI. Extended Blockchain Based Architecture

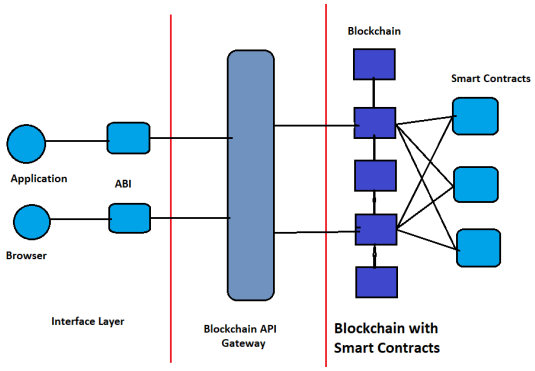

Fig. 3. Extended Blockchain Based Architecture

It contains three layers. It has 1 extra layer from the last architecture. The first is the graphical user interface. The second layer is the blockchain API Gateway like Blockchain workbench API. The third layer consists set of smart contracts deployed on the many blockchains. This architecture is useful when we have a complex system.

VII. Blockchain Implementation Of The Case Study

This section explains the microservices-based system's implementation process. We began with the assumption that Ethereum enables the development of decentralized apps that run on hardware that is linked to the peer-to-peer network. That application works in the Ethereum VirtualMachine and is named smart contracts.

Our goal is to create a remotely callable smart contract Doctor which generates requested data by calling other two remote contracts knowing only their address and their interface. Three smart contracts were written in a single source file that complied with Solidity version 0.5.0 in order to manage the dependencies between element declarations simply.

So we implemented 3 atomic smart contracts for one of each microservices(Docter, patient, Diagnosis). To easily manage dependencies between elements declaration of three smart contracts in a single source code and to reproduce the concept of encapsulation we implemented patient and diagnosis classes as solidity libraries.

The patient library includes variables like ID, Patient_name, Patient_email, and related getters and setters methods. Library diagnosis includes variables like ID, disease_name, disease_descrption, and related getters and setters.

The patient library's behavior represents the microservices that get patients. It takes Patient ID and returns patient data. Similarly, diagnosis behavior represents getting a diagnosis that takes input diagnosis ID and returns set of matching data. Doctor library's behavior knows the URL of the microservices diagnosis and patient and provides a callable submit order function which gives formatted output that includes data related to the specific patient and to the specific disease.

Below are the steps of how the function does work at the time of order creation.

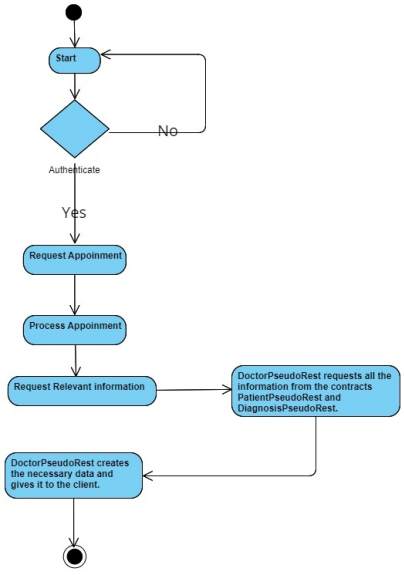

Fig. 4. Activity Diagram

1) A "customer" requests the creation of an appointment from the contract DoctorPseudoRest using the function create order, providing the disease ID, patient ID, and the amount.
2) At this point, the contract DoctorPseudoRest requests all the information pertaining to those specific IDs from the contracts PatientPseudoRest and DiagnosisPseudoRest.
3) In response, the two contracts give the necessary information (a patient object and a diagnosis object).
4) In the contract's last step, DoctorPseudoRest creates the necessary data and gives it to the client.

VIII. Conclusion

In this work, we examined how a collection of Smart Contracts may really be used to replicate a microservice architecture. We examine the architecture of how blockchain-based smart contracts may be used to construct microservices architecture. Additionally, attempt to suggest a new architecture with minor adjustments that could be advantageous. This provides a real illustration of how a current and widely used software paradigm can be readily mapped using a blockchain approach, opening up new prospects for expanding the scenarios where the usage of blockchain and smart contracts might find applications.

References

[1] R. Tonelli, M. I. Lunesu, A. Pinna, D. Taibi and M. Marchesi, "Implementing a Microservices System with Blockchain Smart Contracts," 2019 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE), 2019, pp. 22-31, doi: 10.1109/IWBOSE.2019.8666520.

[2] Roberto Tonelli, Andrea Pinna, Gavina Baralla, and Simona Ibba. 2018. Ethereum smart contracts as blockchain-oriented microservices. In Proceedings of the 19th International Conference on Agile Software Development: Companion (XP '18). Association for Computing Machinery, New York, NY, USA, Article 21, 1–2. https://doi.org/10.1145/3234152.3234190.

[3] G. Destefanis, M. Marchesi, M. Ortu, R. Tonelli, A. Bracciali and R. Hierons, "Smart contracts vulnerabilities: a call for blockchain software engineering?," 2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE), 2018, pp. 19-25, doi: 10.1109/IWBOSE.2018.8327567.

[4] H. Rocha and S. Ducasse, "Preliminary Steps Towards Modeling Blockchain Oriented Software," 2018 IEEE/ACM 1st International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB), 2018, pp. 52-57.

[5] Tonelli, R., Pinna, A., Baralla, G., Ibba, S. Ethereum Smart Contracts as Blockchain-oriented Microservices.

[6] K. Bakshi, "Microservices-based software architecture and approaches," 2017 IEEE Aerospace Conference, 2017, pp. 1-8, doi: 10.1109/AERO.2017.7943959.

[7] R. Tonelli, M. I. Lunesu, A. Pinna, D. Taibi and M. Marchesi, "Implementing a Microservices System with Blockchain Smart Contracts," 2019 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE), 2019, pp. 22-31, doi: 10.1109/IWBOSE.2019.8666520.