Enhancing Resource Management in Cloud with Blockchain for Transparent Recording of Scaling Events

MSc Research Project
Cloud Computing

Ashwin Sabu
Student ID: 23196505

School of Computing
National College of Ireland

Supervisor: Prof. Diego Lugones

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Ashwin Sabu
Student ID:	23196505
Programme:	Cloud Computing
Year:	2024
Module:	MSc Research Project
Supervisor:	Prof. Diego Lugones
Submission Due Date:	12/12/2024
Project Title:	Enhancing Resource Management in Cloud with Blockchain for Transparent Recording of Scaling Events
Word Count:	9490
Page Count:	21

Signature:	Ashwin Sabu
Date:	11th December 2024

Enhancing Resource Management in Cloud with Blockchain for Transparent Recording of Scaling Events

Ashwin Sabu
23196505

Abstract

Cloud computing is a growing field where many services are used by businesses due to its features such as scalability and ease of access to resources on demand. Since the cloud providers manage the resource allocation, transparency in it would be a requirement. Bringing in a decentralized approach would make the system more transparent to the cloud users. This study proposes to build a system that would store the scaling information and the logs that help to identify the scaling requirement to Hyperledger Fabric. To perform this, external monitoring tools are used along with AWS Services such as EC2, SQS, DynamoDB and designed to build a well-performing system under various workloads. Various experiments were conducted to identify the issues in the system and identify the design that could process all the requests successfully. The system's performance outperformed the previous research and showed improved latency and throughput in the benchmarking test performed using Hyperledger Caliper.

1 Introduction

Cloud computing brings in more opportunities for business to focus on their functionalities rather than going through the infrastructure setup by using services from cloud providers such as AWS, Azure and so on. The demand fluctuates quickly in a dynamic environment due to a variety of causes. When focusing on infrastructure scaling, virtual machines such as Amazon EC2 instance can be scaled up with the demand if the architecture is designed accordingly. That is the modern cloud computing brings in the feature of scalability which enables applications to handle various workloads effectively under various situations. As the dynamic scaling has been maintained by Cloud Providers it brings in a need to ensure that the scaled events are securely stored in an immutable storage for operational and auditing needs in the future. Blockchain technology which has been known for storing the records transparently, ensures immutability of data and therefore Hyperledger Fabric which is a permissioned blockchain is used to ensure that all the scaling events are being stored securely. Here the research would not only focus on storing the data to the blockchain but would consider the limitation of previous research works and optimize the performance by configuring the design to limit the data been sent to blockchain. When considering the importance of this research, increased adoption of Cloud services has brought in the need to ensure that the costs of usage are calculated fairly, and this could be understood by auditing the information events from the logs. Traditional logging methods are not secure and could be tampered with by anyone, which leads to risk in the data integrity and compliance. By utilizing this technology all the challenges would be addressed in case of storing the data and usage of tools such as Artillery load testing along with Hyperledger Caliper would enable to identify the performance issues associated with it which would open a wider area to research. Hyperledger Caliper is a benchmarking tool used to measure the performance of Hyperledger Fabric under different use cases and similarly Artillery helps perform load testing on the entire system. A previous research on storing the information in blockchain (Alsharidah et al. (2022)) shows the performance of the blockchain configured using Hyperledger Caliper but does not cover the performance issues of the overall system. This needs to be understood well enough when designing a system so that proper optimization could be performed.

1.1 Research Question

This research would address the following research question:
How can blockchain technology be used to enhance the transparency of automated infrastructure scaling, making it more trustable to the cloud users? Also, what would be the performance impact on recording the logging events on blockchain to the overall system performance?

Addressing this research problem would bring in a number of objectives that need to be understood and followed. The initial goal would be to find the current state on integrating blockchain with cloud computing. According to the analysis a system needs to be developed which would capture the log information and store it in blockchain. Understanding the performance impact would be another primary goal and benchmarking tools should be used to understand the performance impact it could have and solve any issues that would be blocking the functioning. Finally, measure the overall latency, throughput delays which would brief the performance impact it could have when implemented. Here the research would be performed utilizing AWS Services where Hyperledger Fabric would be installed and configured in EC2 Instance and following all the performance testing in the instance. Also, logs responsible for scaling are being analyzed and sent by Node Exporter, a tool by Prometheus that could be used to capture the system metrics. Like other researches, there are some limitations in this research also which have been addressed in the starting stage. An initial thought would be the point of bringing in a monitoring tool like Node Exporter into the research when AWS itself provides services like CloudWatch. Here, the main concern is to validate the scaling actions performed by the cloud provider and using a cloud provider tool would not help reach a conclusion. Also, another limitation is the hosting of Hyperledger Fabric on the cloud provider, this could have been hosted on any other location but at this point of research, importance is given to identify the performance issues it could have rather than focusing on the location of where blockchain is hosted.

1.2 Structure of Work

The paper has been structured to give a detailed analysis of the research performed which would open the path to further improvements. The next section would start with Related Works that would detail the current state of this research were the recent researches are considered and analyzed to develop the action plan. The Methodology section covers how the log information is collected ensuring data integrity has been considered. Following that the design specification shows a low-level architecture on the implemented system which could be used to understand in depth on the research functions. The implementation section discusses more on the steps taken in the implementation such as exploring the different EC2 instances that are been considered in terms of the performance and cost. The Evaluation section details more on the different experiments performed in this research and discusses more on the issues discovered and how they were addressed. Finally, the conclusion section briefs about the entire research, followed by the future works that could be performed on this research.

2 Related Work

Integrating cloud along with blockchain is a field that is been focused by many researchers around various industries especially in areas where transparency is required to make the system more secure, leading to integrity of the data been ensured. As businesses nowadays rely more on cloud services to expand their infrastructure and operation, the need for transparent systems is leading to demand. Blockchain technology, known for storing data on a decentralized platform which would lead to immutability, has been considered as solutions in various research. When focusing more on cloud, projects were undergone that store log information in blockchain which brings more trust to the users. From bringing in transparency to ensuring security for the data, blockchain has been used in various research. Similarly different approaches are taken to audit the data that is stored in the blockchain and in the cloud to ensure integrity. In this section, various studies performed by researchers are being explored to understand this paper's background and necessity.

2.1 Integrating Blockchain for Logging and Auditing

Many existing works focus on using blockchain to store information from the cloud. In a research, it was explored on storing the log metrics of an EC2 instance and decide if the scaling action performed in the cloud is in accordance with SLA as agreed with the Cloud Provider (Alsharidah et al. (2022)). Here using AWS Services, an application is hosted in EC2 instance, which stores the details of the resource usage by AWS services in blockchain. To identify the factors such as CPU Utilization that are necessary for scaling, a monitoring layer is deployed using M3 monitoring tool which ensures real time monitoring of the server. For blockchain, there is a smart contract deployed for auditing and verifying the decisions made by the cloud provider based on the logs received from the monitoring tool. This smart contract is triggered using AWS SNS notification service when auto-scaling action occurs, which then audits the decision that is made by the cloud provider. Even though the research brings in more transparency by storing the logs in blockchain, this is not feasible for projects that could be sending continuous data to the blockchain and the amount of data send could lead to increase in latency. The authors could have more focused on limiting the data that is been send and therefore avoiding a bandwidth issue in future. Similar to the above research, there is a study performed for auditing the data integrity using blockchain, where the aim was to use non-leaf node method (Wang et al. (2024)). Here the goal was to avoid communication costs from the RMHT-based auditing approaches which were used by many organizations to verify data integrity. The author has also performed security analysis to ensure the effectiveness of this approach. In this paper, the authors try to focus more on logging and auditing the data in a secure and efficient method and do not focus on the application side that is from where the data is to be sent. In a similar method researches are also done focusing on improving the system developed where the security of the data is the main concern(Awadallah and Samsudin (2021)). When comparing the papers discussed above, the research by Alsharidah et al. (2022) focused more onto the effectiveness of scaling EC2 instance while the other from Wang et al. (2024) were more concerned with improving the effectiveness of auditing.

Blockchain mechanisms are also used to enhance the security of databases by performing various security operations such as hashing or encrypting the data (Awadallah and Samsudin (2021)). Hashing methods like SHA-256 are used widely in order to ensure that the integrity of the data has not been compromised. In a paper focusing on enhancing relational database security, it was explored on how SHA-256 can be used in the system to enhance security. Here the authors propose two experiments where one is focused on agility and the other on security. SHA-256 was used in both the systems to prevent unauthorized modifications in the database there by enhancing security to the next level. In an another study, a similar approach was taken by the researchers in Cloud SLA verification. Here a system was created where the cloud users and cloud service providers could verify that the SLA has been met or not without the need of trusting each other. This was achieved by storing the evidence of compliance in blockchain and thereby allowing all the participants to verify this information transparently. Here the authors put more focus on data integrity and verification. Similar to this approach, traditional database and blockchain integrated system has been considered in evaluation on a research and here the pros and cons of both were evaluated (Putz et al. (2019)). Also a decision tree was created for users to choose which approach would be suitable in the case they needed.

2.2 Security Enhancement in Blockchain Cloud Environment

Security of the data is the primary concern for exploring new methodologies to enhance the management of data. In a research conducted by Pise and Patil (2021), it was focused on decentralization of cloud storage by integrating blockchain. The main issues which were addressed was the integrity of data and unauthorized access which was approached by decentralizing the cloud over data storage. This approach taken ensures that a single entity like cloud provider has no full control over the data which could reduce the risk of data breaches. Here SHA-512 is been used to store in block to ensure integrity and the integral concept in blockchain that is one block is been linked with the previous block to enhance the security. This reduces the tampering of data as changing a SHA of a block with at the end eventually break the chain. In addition to it AES Algorithm is used in encryption and decryption of data which indeed would help in maintain confidentiality of data as the key would be required to decrypt the contents. The result of combination of both hashing and encryption makes it more secure from being accessed and tampered illegally. This approach could be used in situations where data integrity is to be ensured and followed which is similar in the problem that needs to be addressed. As the use of hashing techniques are used to improve integrity, it has also been used to protect data from various sources such as IoT. In a research that is conducted by Velmurugadass et al. (2021), the data from IoT devices which is been sent to the cloud need to be securely stored. Here the researchers used Elliptic Curve Integrated Encryption Scheme (ECIES) and hashing algorithm which is SHA-256 that focuses on ensuring security of the data that is been send from IoT to cloud. The main objective addressed would be how software defined networking (SDN) and blockchain can be integrated to ensure secure data flow in cloud and IoT environments. SDN would be used separate the network control and data planes whereas blockchain is used to store the details of IoT data transfers which makes sure that the records are not been tampered. Here the researches uses SHA-256 to generate cryptographic hashes that is been stored to blockchain, this indeed would reduce the amount of data that is been send to blockchain. This project performed by Velmurugadass et al. (2021) focuses more on securing cloud based records and by adding ECIES, an additional layer of protection is added which indeed could be beneficial in handling sensitive data. Similar to this research, there are other papers that uses same approach but utilizing chaotic elliptic cryptography (MICEC) to enhance security in the cloud. MICEC framework has three main stages such as authentication protection, ownership protection and identity mapping validation.

When emphasizing on bringing in security using blockchain, Hyperledger Fabric (HLF) is been used to enhance security. As discussed in the introduction, HLF is used to securely store the data in blockchain and unlike Ethereum it is much better in terms of performance, scalability, and security. In a study performed by Tunstad (2019), the data such as updating a content, timestamps which is been generated from the application on a distributed system are stored on HLF. As HLF is a permissioned blockchain which requires that the users much be verified and trusted to perform any operation, it increases the level of security. The smart contracts or chaincode in HLF are able to manage the storage and retrieval of large amounts of data. The papers discussed in this section focus on ensuring data security and integrity in areas such as cloud storage, IoT cloud data transfer and data handling in distributed systems. Every study has its own pros and cons, implementing a system with better performance and security should be the primary concern.

2.3 Blockchain frameworks and performance in cloud applications

Applications that are hosted in cloud need to be transparent enough in terms of the resource usage from which the billings are calculated. Also, applications that handle sensitive information need to be placed in such a manner that it is very much secure from attacks. Healthcare is one such field where sensitive information needs to be handled securely and there is much ongoing research in it and Hyperledger Fabric plays a significant role in most of the projects because of its benefits. Analyzing the performance of such a system using relevant metrics is necessary in the ongoing research. One such research focuses on measuring the performance of a health care system integrated with Hyperledger Fabric (Al-Sumaidaee et al. (2023)). Here, the application's objective would be to streamline medical data sharing, reduce operational delays, and improve the security and availability of patients' data. The authors created a test network between two health care institutions and performed several testing to evaluate the performance using Hyperledger Caliper, which is a benchmarking tool to evaluate the performance. The three main metrics focused on the system designed are throughput, latency, and transaction processing. The results show that when increasing the transaction from 1000 to 15000, the throughput improved while the latency remained stable. Also, a higher transaction per second resulted in an improved performance but increasing the number of workers caused network overload and reduced performance efficiency. This research shows the impact of the application when high volume transactions occur in the system.

In similar research on healthcare applications focused on utilizing hashing mechanism to improve the integrity of the data (N. and Thippeswamy (2021)). The implementation discussed in the paper proposes a framework that addresses confidentiality and security of sensitive data especially in scenarios where the data is across multiple healthcare providers, and which is stored in the cloud. Here, like the papers in section 2, the authors implemented the system where instead of storing the records in blockchain, they store their hash values. This ensures that the data could not be altered by any attackers and if any changes made would lead to flagging an alert immediately. Also, an access control policy has been implemented along with the smart contracts or chaincode for ensuring only authorized people can access.

2.4 Hyperledger Fabric Performance

Recent studies around measuring the performance of Hyperledger Fabric (HLF) are undergone following the factors such as throughput, latency and scalability. The selection of the programming language used to configure the system is a factor that affects the overall performance and in that according to Foschini et al. (2020), Go Programming language performs much better than Java or Node.js in terms of the execution time and overall transaction efficiency. Throughput is a critical factor in determining the performace of blockchain, in order to improve that Thakkar et al. (2018) reduced the block size by sending limited data and using parallelized endorsement verification. Also further researches show that batching of the transactions and the number of endorsement peers influence the end to end latency which shows a trade-off between scalability and performance (Shalaby et al. (2020)). Another factor would be to identify stable version of HLF, where most used versions v0.6 and v1.0 are compared and identified that v1.0 has an improved latency and throughput due to the architectural enhancements on the ordering services and endorsement policies (Nasir et al. (2018)). Even though the performance issue are considered in various researches, a common issue is with the scalability as a challenge.

2.5 Comparison on Previous researches

In this section, major works that relate to this research would be summarized and understanding these works would help to get a detailed idea on how this research is performed including the reasons behind adopting alternative measure to increase the performance. In table 1, the comparison starts with the first row on the current research followed by subsequent research which shows their objectives and blockchain type used.

Table 1: Comparison of Related Works Table

Reference	Objective	Blockchain Type	Use of Smart Contract
This research	Store log and scaling events in blockchain for transparency	HLF	Store the hashed data
Alsharidah et al. (2022)	Blockchain based autoscaling verification for transparency	HLF	Store the logs and validate CSP autoscaling decisions
Wonjiga et al. (2019)	Ensure data integrity by blockchain	HLF	SLA Compliance verification
Awadallah and Samsudin (2021)	Secure relational database in cloud using blockchain for immutability and access coontrol	HLF	Store the timestamp
Pise and Patil (2021)	Data security in cloud storage using SHA-512	Siacoin / Filecoin	Store hash value

In this section various research on storing information in blockchain securely has been highlighted. Also have seen how hashing could be used in Databases to be incorporated in limiting the amount of data that is been sent to blockchain. At last research on the performance of Hyperledger fabric on the performance and challenges involved such as scalability are mentioned.

3 Methodology

Many researchers use blockchain as a key technology in various research to bring in transparency to a system as the immutable feature of blockchain ensures integrity of the data. As discussed in the last section, Alsharidah et al. (2022) has implemented a system which captures the scaling information to Hyperledger Fabric, this indeed results in more bandwidth due to the amount of data that results in high latency and throughput which could slow down the overall process. A first approach to solving this issue would be to reduce the amount of data sent to blockchain, but the data stored should contain the vital information necessary for auditing. As shown in figure 1, the approach would be to avoid storing the entire data to blockchain, instead store the data to DynamoDB and utilize blockchain to verify the integrity of data stored in DynamoDB. In this section, the methodology of bringing in transparency of cloud scaling events is focused on which indeed could be used in the calculation of billing cost or to understand whether the scaling event was necessary by capturing metrics such as CPU usage.

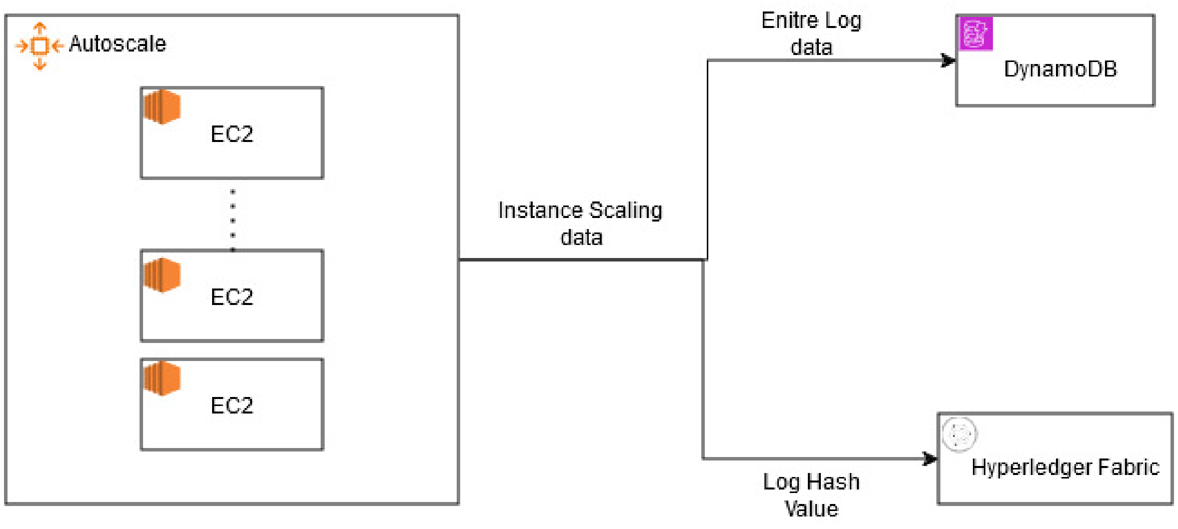


Figure 1: Data flow of scaling events

3.1 Data Collection Process

An integral part of the research is the information that is required to identify the need for performing a scaling operation in the cloud. The process starts from the EC2 instance where Prometheus node exporter is installed and gathers the metrics. Here installing this agent in the EC2 instance helps to gather CPU metrics information which could be one of the factors that determines whether the instance need to be scaled up or not and indeed gathering this information would help in determining if the scaling decision were correct. The log data from the node exporter is send to DynamoDB where the entire log data is stored along with a unique ID which differentiates each record as mentioned in table 2. At the same time, the data is sent to Hyperledger Fabric in the form of ID (which is generated earlier), SHA-256 Value of the entire log data. Similar to sending log data, scaling events also need to be captured and stored, for that Amazon Event Bridge is used. This service captures the scaling events which then are processed by lambda and send to DynamoDB and Hyperledger Fabric. Therefore, there are two approaches used to store the data, the first approach would be used to send the log data of the system metrics captured by the Node Exporter directly to the required locations and the other would be to capture the scaling events by Amazon Event Bridge, which then is processed by lambda and stored in blockchain and DynamoDB.

3.2 Tools and Technologies Used

In this research, AWS Services has been widely used to store logs in Hyperledger fabric. AWS EC2 Service has been used to host the application from which logs are generated and another EC2 instance is used to install and configure Hyperledger Fabric which would be used to store the log information. Since Hyperledger Fabric is a permissioned and private blockchain, as discussed in the introduction, this would be used to store the log data in a secure and immutable way. DynamoDB is another service which is used to store logs. DynamoDB would have the entire detail on the data from the log information that is it would contain information on the CPU usage of the instance in a fixed time and the scaling events would be logged into the DynamoDB which would help in the

Table 2: Data Storage Fields in DynamoDB and Hyperledger Fabric

Storage System	Field Name	Description
DynamoDB	ID	Unique identifier for each record
	Details	Complete log information, including metadata such as timestamp and instance information
	HashValue	SHA-256 hash of the 'Details' and 'ID' fields for ensuring data integrity
Hyperledger Fabric	ID	Unique identifier for each record, same as in DynamoDB
	HashValue	SHA-256 hash of the 'Details' and 'ID' fields, stored securely on the blockchain

decision making. AWS Lambda is another service which has been created in a Node.js environment and is used to store the information in DynamoDB and blockchain. Since AWS lambda is a serverless service, the management of the service is not a challenge here and therefore the configuration could be performed easily. Node exporter is another tool that has been used to gather the CPU Usage details of the instances, as discussed before it is an agent used in Prometheus to capture the metrics of the system. Using all these services the implementation of the proposed system was performed. To measure the performance of the Hyperledger fabric, which is deployed on EC2 instance, Hyperledger Caliper which is a benchmarking tool has been used under various workloads.

3.3 Data Integrity and Verification Process

Ensuring the integrity of the data is the primary aim in this project. Here the log details along with the unique ID is been hashed using SHA-256 algorithm and this hashed value is stored to a variable. When sending the record, DynamoDB received the entire log details along with the hashed value, whereas Hyperledger fabric receives only the ID and SHA-256 value. In order to ensure that the data stored is DynamoDB is not tampered, the hash value of the record could be verified. For that the first step would be the recalculate SHA-256 value stored in DynamoDB, since altering the record in the database will not change the SHA value. Following this step, the calculated SHA value and the value in the blockchain needs to be verified by using the unique ID present in it, this ID acts as the primary key and is used to get the record from the Hyperledger Fabric. This would ensure that the data in the DynamoDB has not been altered and in this way the amount of the data send to blockchain is reduced, thereby resulting in the system performance improvement.

The methods chosen in this research focuses primarily on bringing transparency by ensuring the integrity of the data that is been stored. In this process, the extracted data is been stored in location in different forms which ensures that the data could be integral. Storing the complete record in blockchain could be performed but it wont be efficient since it would take high latency to process it. When considering a system which generated lots of data in a second, storing large amount of records in blockchain is not feasible. Therefore the methodology chosen ensures that the performance of the system is not affected if the amount of the data been send is large.

4 Design Specification

Design of the system focuses more on increasing the performance of the overall system by reducing the amount of data that is been transmitted especially to the blockchain. Here there are three main phases, and each phase would contain proper design aspects which ensures that the data are stored securely, and the integrity is maintained. The requirement considered in this research would be to increase the overall performance of the system and therefore optimize the areas where more processing takes place. In figure 2, architectural diagram of how the system performs could be seen and flow of the data from the EC2 instance to DynamoDB and Hyperledger Fabric. Here data is been sent through two channels, the first one is the metrics from the node exporter and the second is the events that is been captured by the AWS EventBridge which is sent to lambda. The metrics captured from node exporter is taken in a fixed interval of 5 seconds and is send directly to DynamoDB and Hyperledger Fabric. This ensures that the events are captured continuously and from the data stored, analysis could be performed to identify if the scaling action was required by the cloud provider. Similarly recording the scaling action is another primary requirement and is satisfied by using AWS Event Bridge. AWS EventBridge is configured to capture the instance creation and termination which send the information to AWS lambda as before. Once processing is completed in lambda, the data is stored to the database and blockchain.

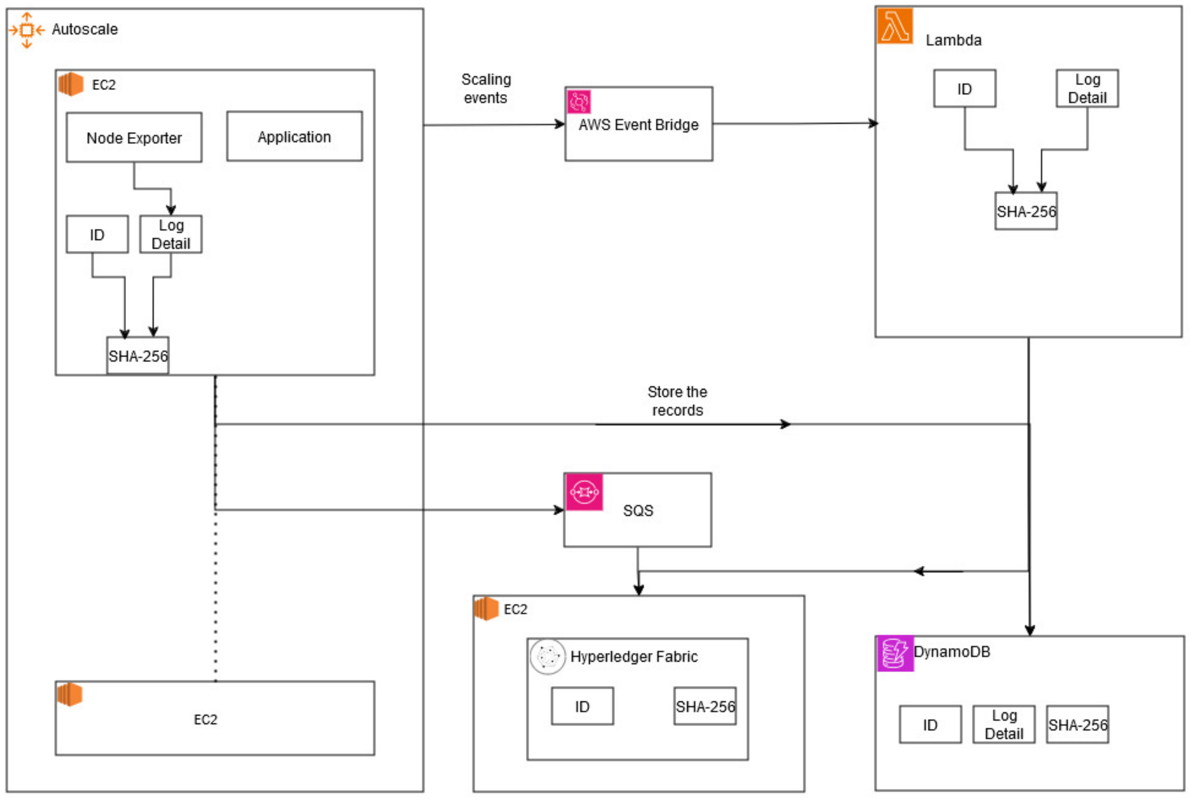
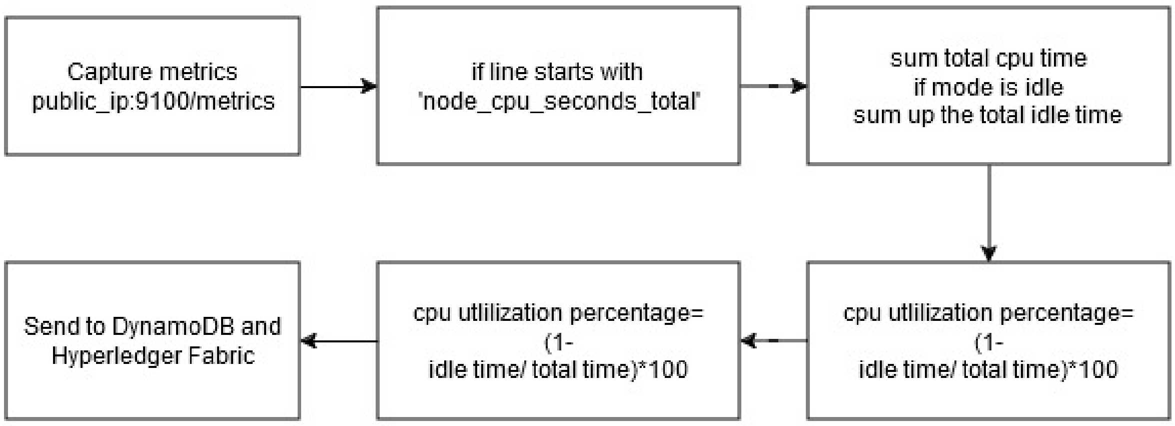


Figure 2: Architectural diagram

4.1 System Components

Each component in the system contributes to storing of information and it is been ensured that only required amount of information is taken discarding the remaining. When considering for node exporter, it generates a multiple statements in which all are not needed to be stored and therefore python script has been used to capture only the necessary events. Similarly AWS Event Bridge comes with multiple functions and is only used in this project to record the scaling event information.



Capture metrics public_ip:9100/metrics
if line starts with 'node_cpu_seconds_total'
sum total cpu time if mode is idle sum up the total idle time
cpu utlilization percentage= (1- idle time/ total time)*100
cpu utlilization percentage= (1- idle time/ total time)*100
Send to DynamoDB and Hyperledger Fabric

Figure 3: Node exporter data flow

4.1.1 Application in EC2 Instance

An application is hosted in EC2 instance where the details of the log from DynamoDB would be visible to the user. Express framework has been used to deploy the application and it consist of a table which shows the last 10 records of the events that has occurred. Here along with the application node exporter is also been used which capture the system metrics. In figure 3, shows the flow of how the metrics are been calculated and stored. As discussed above, python script has been deployed to capture the events from the node exporter which is running on the URL publicip:9100/metrics. The script goes through all the log lines and extract the lines that start with 'node_cpu_seconds_total' and based on that the CPU usage percentage is been calculated.

4.1.2 Processing of Data

Here the log information received from the Node exporter needs to be processed before storing it. Therefore, the information received would be merged into a single variable called details and a unique ID would be generated. Then using Crypto library SHA-256 function is performed on it to hash the details which would now be considered as the third attribute when storing it to DynamoDB. Necessary permissions are attached with the lambda function and EC2, which enables the sending of records to DynamoDB whereas in Hyperledger Fabric, using API the data would be sent. Since it has been hosted in EC2 instance, the post request is sent through API using axios function. When performing some test cases on the lambda function to ensure that the transmitting of data was possible, it has been seen that in some occurrences the lambda function has failed due to the timeout issue. This was resolved initially by increasing the default timeout from 3 to 10 seconds, due to the time taken to transmit the data to Hyperledger fabric. In proceeding further SQS has been added to the design which would ensure that the requests sent to Hyperledger Fabric would not fail and therefore ensure that all the requests are being processed.

4.1.3 Chaincode Development and Deployment

Setting up Hyperledger Fabric (HLF) to store the hashed value is another crucial factor. For that after installing HLF, chaincode was created to record the ID and SHA from the lambda. ID would be considered as unique and will not accept any duplicate entry to it. Once the configuration on HLF was performed, the testing was completed locally to ensure that it would be able to store the record in it and ensured to perform the testing through the EC2 instance public API to ensure lambda would be able to communicate to it. Since the configuration of instance was t2.small in the first stage, there was performance issues as the system properties where low to handle the testing performed on different workload, after changing the instance type to a higher configuration it worked as expected. Like the application running on EC2 instance, here also Express framework has been used since lambda has to communicate with HLF to store that data.

4.2 Load Testing and Optimization Using SQS

To ensure that the design followed would be able to handle different load as the time demands and for that load testing was performed using an open-source tool called Artillery. Here following the installation of the tool on the EC2 instance where the application is hosted, a different load was given in the configuration script. When the testing was performed with initial configuration of 5 transactions per second for a total duration of 30 seconds, it was noted that the system was able to handle up to 10 transactions per second and after that many requests returned 500 errors. This was because lambda has a request limit of 1000 concurrent executions per account and here when a transaction fails it would retry the request which would reach its limit where the requests would end up with a 500 error. Therefore, the design has been changed to send the log metrics from EC2 to directly where the data needs to be stored instead of sending it to Lambda and in this case processing of the data which involved calculating the hash value was performed on EC2 itself instead of Lambda. But the scaling events were processed by lambda as it would not exceed the lambda processing limit since the scaling events do not occur frequently as of the log generation. Another issue which was discovered is that the Hyperledger fabric could fail after a certain limit, and it is always necessary to ensure that all the requests sent are processed successfully. To resolve this issue, AWS SQS has been used which would handle the requests as the queue, thereby ensuring that all the transactions are processed. For that, SQS was created and instead of redirecting the requests from EC2 to Hyperledger Fabric, it was sent through SQS which would queue up the data and ensure it is stored. This ensured that all the requests were forwarded without losing any requests. One of the challenges faced here was that SQS would only accept request AWS signed requests and when sending the requests, it was getting failed due to this reason. To resolve this issue, AWS Artillery plugin was used which ensured that the signatures were generated from a trusted source. This testing was performed starting with 5 transactions per second to 100 transactions per second for 60 seconds.

5 Implementation

Enhancing performance would be the primary consideration when developing a system to handle large scale deployments. Here the implementation section focuses on the different phases through the research where there where was performance issues found and resolved. The configuration which are done in the system takes control on how much resources it can take and in cloud it can be handled much easily without having much impact on the overall system performance. The development approach in this project focused on reducing the workload on Hyperledger Fabric which indeed would increase the overall performance of the system.

5.1 Setting Up the Cloud and Blockchain Infrastructure

AWS Cloud services was used in the setting up of the entire infrastructure used in the project. Services such as EC2, Lambda, Event Bridge are used to perform the research where Hyperledger Fabric (HLF) was installed on EC2 service. Here two EC2 instances where initially used, where one instance in which application is installed is in the auto scaling group that scales up or down based on the traffic and other instance where HLF is installed. The EC2 instance where the application is installed is configured to use the lowest configuration which is the t2.micro since for research purposes this would fulfill the need. The instance on which HLF is installed uses a higher configuration of c5.9xlarge. This higher instance was chosen to perform testing by performing benchmarking tests with lots of transactions and therefore configuring a system with a smaller instance like t2.small would not satisfy the requirement. Considering reducing the cost during the research, initially t2.small has been used to install and configure the system and once everything was working as expected, an AMI of the instance was created and launched an instance with the higher configuration to perform only the testing. The following table 3, shows the different configurations that are used in this research setup and how they differ in terms of the performance.

Instance	**vCPU**	**Memory (GB)**
t2.micro	1	1
t2.small	1	2
c5.9xlarge	36	72

Table 3: Specifications of AWS Instances

The HLF configuration was performed by setting up an orderer and one peer in an organization with the required ports for communication. Channels where also setup to communicate securely with the system from outside the instance, this includes the user being able to submit transaction, and for the research purpose it was made open to all without requiring any credential to submit the transaction as the aim was to focus more on the performance of the system. Following the configuration, chaincode was created and deployed to the channel to accept the ID and SHA-256 value which would be used to link with the records in DynamoDB.

5.2 Configuration for Data Processing

As discussed earlier, the records are stored from EC2 instance and lambda. In this research, the primary focus would be given on storing the logs from EC2 instance as this could generate more number of requests with the increase in number of instances. Incase of the EC2 instance, permission to access the DynamoDB should be provided in the IAM role attached with the instance and using AWS Configure command, IAM credentials need to be configured to enable transmitting of the requests. Next would be to capture the events from AWS Event Bridge, where the Lambda function is setup in a Node JS environment and following the creation, the package is uploaded in a zip format which contains all the libraries that are required for the lambda to function as needed. Here the code is been written in an index.js file where it contains creating a client to Dynamo DB through which the data could be send and to store in blockchain axios is used where the data is send through Express API. As discussed in the design section, one of the challenge faced during this phase was that at certain testings, lambda was not able to complete the transaction successfully since HLF was not responding to the requests in the default timeout of 3 seconds. Even though changing the timeout did not permanently resolve the issue as HLF is expected to handle lots of request and therefore increasing the resources of the EC2 instance from t2.small have shown better performance.

5.3 Monitoring the Instance

Node exporter which is one of the agents is used in Prometheus to capture the system metrics that is been installed in EC2 instance. It is installed and configured on port 9100 where the metrics of the system would be available and as discussed in figure 3, the metrics are captured from the EC2 and send to lambda. Here the python script is executed every 5 seconds from which the system behavior could be captured. An alternative and more simpler method would be to directly take the metrics from cloud watch but using a tool from the Cloud Provider would not completely show the reason for scaling and therefore an external tool is been installed to it. For the research purpose, only CPU Usage is been calculated and in future it could be considered to capture more metrics that would bring out more relevant information on the instances.

5.4 Testing and Validation

In order to ensure that the system is working as expected, testing was conducted in multiple ways to ensure that it would be able to handle different workloads. Lambda function was tested individually with different inputs to ensure it is able to store the data in the necessary formats when the scaling events are send from AWS Event Bridge. As discussed earlier, Hyperledger Caliper has been used to perform benchmarking assessments on the overall system. Before initiating the testing, the total number of transactions was set to 1000 and for each test case the number of transaction per second (tps) increased by 100. Along with that, the worker nodes were also changed from 1 to 5, and the system's performance was observed with different send rates. For each test case the throughput and latency was measured and compared with a similar approach taken by Alsharidah et al. (2022) in order to identify the difference in performance.

Table 4: Performance Testing with Hyperledger Caliper

Round	**TPS**	**Worker Nodes**
1	100	1 / 5
2	200	1 / 5
3	300	1 / 5
4	400	1 / 5
5	500	1 / 5

Similarly to identify the performance of the system on various load conditions, different configurations as mentioned in Table 5 is used. Necessary improvements are made inorder to ensure that the system would handle the load effectively.

Table 5: Testing performance with Artillery

Round	**TPS**	**Total Duration (seconds)**
1	5	60
2	10	60
3	20	60
4	50	60
5	100	60

5.5 Enhancing Chaincode Efficiency

The chaincode in this research ensures that the data are been stored to Hyperledger Fabric securely. Go programming language was used in the development to ensure efficiency in handling requests. Here as discussed earlier the initial thought in optimizing the performance would be to minimize the data and therefore only ID and SHA has been used which indeed would result in a improved throughput. Likewise another concern was to ensure that whether the system would be able to handle multiple concurrent transaction, and for that Hyperledger Caliper has been used and shown that a high number of transaction where able to be handled due to its lightweight implementation. At last by marshaling the data to a JSON format in the chaincode reduced the overhead during the write operation. Even though the blockchain handles a high number of requests, it could be improved much further to decrease the overall time of the system. When calculating the latency of the system, the majority of the time goes towards storing the data in Blockchain and the latency in storing to DynamoDB is really low. Similary batch asset creation could be focused further which could reduce transaction overhead during the peak times.

6 Evaluation

Analyzing various approaches on performance improvement helps to identify the best possible method to be used that covers most of the limitations that could be encountered. In this section, the results of the implemented methodology would be analyzed and compared to the previous work performed (Alsharidah et al. (2022)). Storing log information to blockchain brings in transparency but there are various challenges to be addressed.

This research has been developed using AWS Cloud Services and one of the challenges addressed was to overcome the limits on services imposed by AWS. One such limitation is the request limit in Lambda service on a basic AWS account. This would be discussed further in the following subsections on the experiments performed. Also as discussed earlier, the experiments revolve around the different approaches on storing the system metrics in blockchain and the scaling events are primarily processed and stored using AWS EventBridge and Lambda services.

6.1 HLF Performance

Here using Hyperledger Caliper the performance of the blockchain is measured to identify how the system would react on a increasing traffic rates. Here the metrics that would be measured is the throughput and latency under various workloads. As discussed, the benchmarking test is conducted from 100 TPS to 700 TPS (Round 1 - Round 7) with number of worker node set to 1 and 5 respectively for both configurations. When comparing the results from the research performed by Alsharidah et al. (2022), it is seen in 4 that the throughput is almost the same with a slight difference showing a lower value, but at the same time 6 shows that the latency is much lower where in Alsharidah et al. (2022) was 3.63 seconds for 700 transaction per second where as in the current research it is 0.11 seconds. This is primarily due to the reduction in the amount of data that is been processed and therefore the latency decreases. For the second configuration in fig 5with number of workers set to 5 shows much better performance to the previous research. When considering the case of round 7 where the configuration was set to aim 700 TPS, the throughput was 283 TPS when compared to 240 TPS (Alsharidah et al. (2022)). Similarly the latency also results in a much lower value which shows the performance improvement.

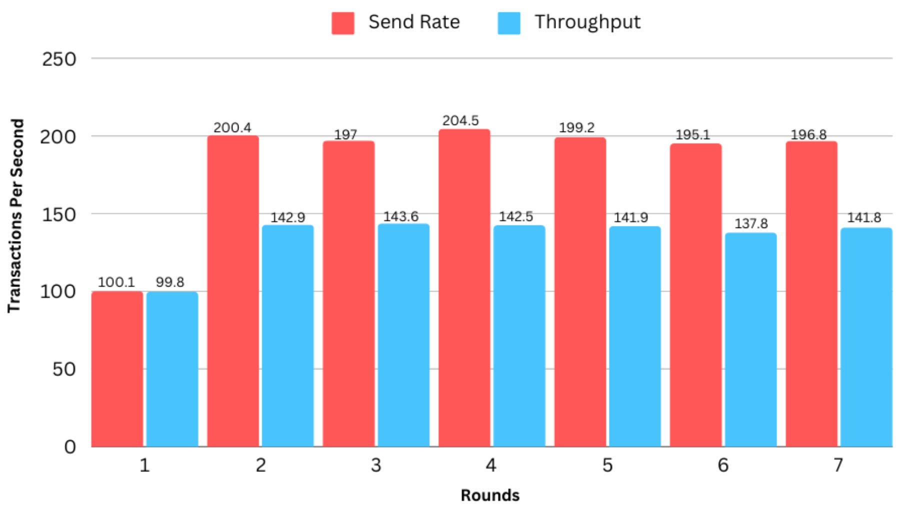


Figure 4: Throughput and send rate with 1 workers

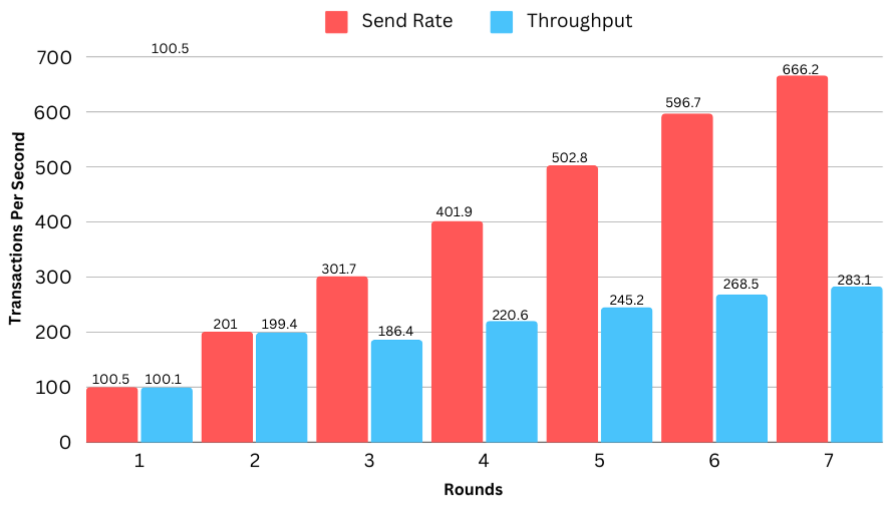


Figure 5: Throughput and send rate with 5 workers

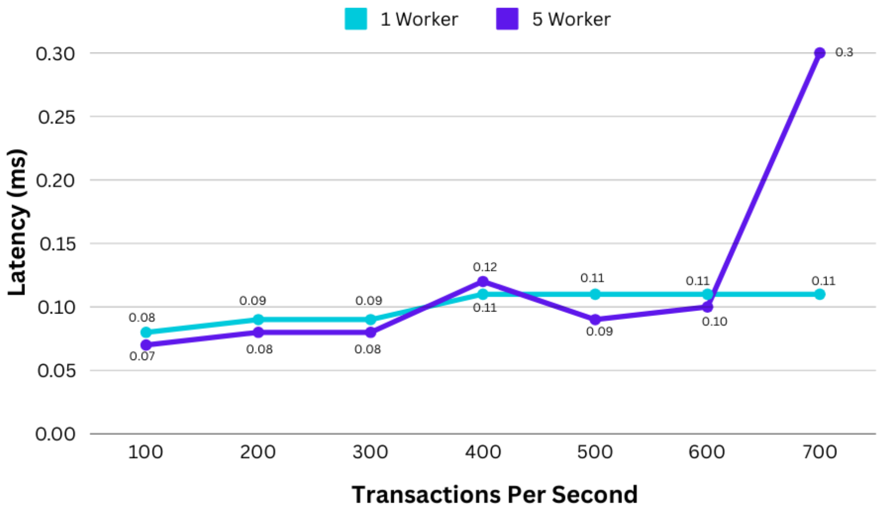


Figure 6: Latency in two configurations

6.2 Experiment 1 : Integration with Lambda

The first phase on the implementation was to analyze the performance of the system with lambda in it. Here EC2 would system metrics collected from Node Exporter as a request to lambda and after processing, it would send the requests to store the data in Hyperledger Fabric and DynamoDB. When performing load testing on this experiment, it was noted that there are performance issues when sending above 10 requests per second. The below table 6 shows the details of the load testing performed using Artillery Testing Tool and it has been seen that from 20 transactions per second the requests are failing and as the number of requests get higher, the percentage of failed requests is increasing. Upon investigating on this it was found that the limit was on the lambda that the Invocation requests on a single Lambda function per Region is 10 (that is lambda accepts only upto 10 requests per second), therefore the requests send to lambda was failing when it went above the limit.

Table 6: Performance Metrics for Different Request Rates

Requests/Second	**Success**	**Failed**
5	300	0
10	600	0
20	1103	97
40	1823	390
50	2476	524
100	801	5199

Also latency of the overall system was identified and it showed that an average time of 3000 ms is required to process a single request. Here the processing time of Hyperledger Fabric is 2950 ms and DynamoDB is 50 ms. Analyzing the performance of HLF it is understandable that the increase in the latency is due to how Express API handles the requests and therefore resulting in an increased latency. When looking into the Lambda limitations, this could be further solved by avoiding lambda in the next experiments as removing lambda and sending directly to the required destinations. The main purpose of lambda in the design is to handle the processing of data such as calculating SHA value and also for recording the scaling events of EC2 instances.

6.3 Experiment 2 : Integration without Lambda

Here when removing lambda from the design, the processing of the data takes place in EC2 and following that it is send to DynamoDB and Hyperledger Fabric. But when trying to overcome the limitation of lambda it is seen that the HLF is not able to process transaction after a limit showing a timeout error. This could be primarly due to the configuration on accepting the data or the issue with Express API in handling the requests because in fig 4 and fig5 it could be seen that the testing using Hyperledger Capliper on the system performed well all the rounds without any error.

Table 7: Performance Metrics for Different Request Rates

Requests/Second	**Success**	**Failed**
5	300	0
10	600	0
20	1200	0
40	1800	0
70	4200	0
80	427	4373
100	1187	4813

In table 8, it is seen that from 80 request per second there is an increase in the failure rates and as discussed above the current configuration is facing errors when the number of requests increased. Therefore the next experiment was to bring forward a solution to ensure that the there all the data are been saved in blockchain even at a peak condition. A useful metrics which could be calculated from this results is the throughput delay product which shows how much transactions is 'in transit' in the network at a given moment of time. Here considering the throughput to be 70 TPS and the round trip delay to be 3000 ms or 3 seconds, throughput delay product would be 210 transactions.

6.4 Experiment 3 : Integration with SQS

In this section, AWS SQS service is been integrated with the design to handle all the requests send to Hyperledger Fabric. After the experiment performed above it has been identified that requests above a certain limit is not been processed by HLF and resulted in not storing the information. This issue does not affect DynamoDB as it was able to accept all the requests within the current configuration. Therefore the requests send to HLF are send to a queue first, that is AWS SQS. Therefore it is ensured that all the request get stored in HLF even through it might take some time. Here the objective is not to reduce the processing time but to ensure that the data gets stored and solid for auditing purposes. Therefore the final experiment shows on processing the data on EC2 and sending directly to DynamoDB. Incase of the HLF, the requests are send to AWS SQS from there it is send to blockchain. When analyzing the performance using load testing it is seen that all the requests are successful as all the requests are in the queue.

Table 8: Performance Metrics for Different Request Rates

Requests/Second	**Success**	**Failed**
5	300	0
10	600	0
20	1200	0
40	1800	0
50	3000	0
100	6000	0

In all the three experiment the latency of processing a single request remain the same which is 3000ms (on average). The services added in the design ensure that it is able to handle requests at a high traffic level when considering many instances which send log details on a short duration interval. From the benchmarking results of Hyperledger Caliper it is understandable that the current system performs well under various workloads and performs much better compared to the previous researches. When considering the cost of hosting, it would be moderate since the number of AWS services used in the final experiment is limited. Here the majority of the cost goes to AWS EC2 instance which would be running continuously. Since the application is hosted on a t2.micro instance, cost would be less for this research but when its deployed for a large scale users, the instance type needs to be changed to a higher configuration. For HLF, the instance type is c5.9xlarge which is a large instance that could cost near to a thousand dollar per month. For SQS the costing would be based on the requests received and the first one million request would be free of cost every month. Therefore integrating transparency to a system brings in a considerable cost that could be used to ensure data integrity and transparency.

7 Conclusion and Future Work

In this research performed the primary objective of bringing in transparency by storing the scaling event and system metric information in Hyperledger Fabric is been addressed. The design of the system resulted developed by performing various experiments resulted in better performance compared to the previous researches conducted. Here Node Exporter was installed as an alternative to AWS CloudWatch to capture the system metrics that could be used in the auditing to identify if the scaling action performed by the cloud provider was in accordance with the SLA. This information along with the scaling events that is been captured is stored in Hyperledger Fabric and DynamoDB. Bringing in DynamoDB reduced the amount of data that is send to blockchain and there blockchain is used only to ensure the integrity of data stored in DynamoDB. Load testing was performed individually on the blockchain and the overall system to identify the performance it could bring under various loads..

This research opens a wider area for future work to proceed from the auditing part of the stored data in blockchain. Also there could be configuration improvements done on the Hyperledger Fabric to reduce the latency more, as a high percentage of overall time of the system to process a request is on the blockchain to process. Similarly in commercialization of the project, Hyperledger Fabric could be installed and configured outside the cloud provider to enhance more transparency further.

References

Al-Sumaidaee, G., Alkhudary, R., Zilic, Z. and Swidan, A. (2023). Performance analysis of a private blockchain network built on hyperledger fabric for healthcare, *Information Processing Management* **60**(2): 103160.
URL: *https://www.sciencedirect.com/science/article/pii/S0306457322002618*

Alsharidah, A. A., Barati, M., Bergami, G. and Ranjan, R. (2022). Cloud auto-scaling auditing approach using blockchain, *2022 IEEE/ACM 15th International Conference on Utility and Cloud Computing (UCC)*, pp. 391–398.

Awadallah, R. and Samsudin, A. (2021). Using blockchain in cloud computing to enhance relational database security, *IEEE Access* **9**: 137353–137366.

Foschini, L., Gavagna, A., Martuscelli, G. and Montanari, R. (2020). Hyperledger fabric blockchain: Chaincode performance analysis, *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, pp. 1–6.

N., N. and Thippeswamy, K. (2021). A framework for secure ehealth data privacy preserving on blockchain with sha-256 in cloud environment, *Turkish Online Journal of Qualitative Inquiry*, Vol. 12, pp. 3152–3167. Accessed: 23 October 2024.
URL: *https://research.ebsco.com/linkprocessor/plink?id=27035d0f-0aac-30f0-bdf0-98537f5880a7*

Nasir, Q., Qasse, I. A., Abu Talib, M. and Nassif, A. B. (2018). Performance analysis of hyperledger fabric platforms, *Security and Communication Networks* **2018**(1): 3976093.
URL: *https://onlinelibrary.wiley.com/doi/abs/10.1155/2018/3976093*

Pise, R. and Patil, S. (2021). Enhancing security of data in cloud storage using decentralized blockchain, *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, pp. 161–167.

Putz, B., Menges, F. and Pernul, G. (2019). A secure and auditable logging infrastructure based on a permissioned blockchain, *Computers Security* **87**: 101602.
URL: *https://www.sciencedirect.com/science/article/pii/S0167404818313907*

Shalaby, S., Abdellatif, A. A., Al-Ali, A., Mohamed, A., Erbad, A. and Guizani, M. (2020). Performance evaluation of hyperledger fabric, *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*, pp. 608–613.

Thakkar, P., Nathan, S. and Viswanathan, B. (2018). Performance benchmarking and optimizing hyperledger fabric blockchain platform, *2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pp. 264–276.

Tunstad, P. (2019). Hyperprov: Blockchain-based data provenance using hyperledger fabric, *Hyperprov: Blockchain-based Data Provenance using Hyperledger Fabric.*
URL: *https://munin.uit.no/handle/10037/15780*

Velmurugadass, P., Dhanasekaran, S., Shasi Anand, S. and Vasudevan, V. (2021). Enhancing blockchain security in cloud computing with iot environment using ecies and cryptography hash algorithm, *Materials Today: Proceedings* **37**: 2653–2659. International Conference on Newer Trends and Innovation in Mechanical Engineering: Materials Science.
URL: *https://www.sciencedirect.com/science/article/pii/S2214785320364014*

Wang, C., Sun, Y., Liu, B., Xue, L. and Guan, X. (2024). Blockchain-based dynamic cloud data integrity auditing via non-leaf node sampling of rank-based merkle hash tree, *IEEE Transactions on Network Science and Engineering* pp. 1–12.

Wonjiga, A. T., Peisert, S., Rilling, L. and Morin, C. (2019). Blockchain as a trusted component in cloud sla verification, *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing Companion*, UCC '19 Companion, Association for Computing Machinery, New York, NY, USA, p. 93–100.
URL: *https://doi.org/10.1145/3368235.3368872*