New techniques to integrate blockchain in Internet of Things scenarios for massive data management.

by

Cristhian Martínez Rendón

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science and Technology

Universidad Carlos III de Madrid

Advisor: Prof. Jesús Carretero

2023

A mis padres y abuelos,
por ser mi inspiración constante
en este camino hacia el logro de mis metas.

"Allí donde haya alguien luchando por asentarse en algún lugar,
o por un trabajo decente o una mano amiga,
allá donde haya alguien que luche por la libertad,
mira en sus ojos mamá porque allí estaré yo..."
Las uvas de la ira (1940)

ACKNOWLEDGEMENTS

Resulta difícil estar lejos de casa. Sin embargo, es alentador y motivador el hecho de saber que a pesar del tiempo y la distancia, la familia y seres queridos perduran. Esta tesis, es un reflejo del amor y apoyo incondicional que ustedes me han brindado a lo largo de mi vida. Gracias a cada uno de ustedes por su voz de aliento y constante apoyo.

A mis padres, gracias por los valores inculcados, enseñarme a perseverar en los momentos difíciles, brindarme las herramientas necesarias para enfrentar la vida y ser el mejor apoyo donde quiera que esté. A mi madre, por ser mi principal fuente de inspiración, gracias por tu entrega y amor incondicional. A mis abuelos por su eterno amor, en especial mi abuelo, que aunque tu enfermedad se ha llevado gran parte de nuestros buenos recuerdos, conservo como un tesoro cada uno de tus consejos e historias que siempre me contabas. Tu nobleza y bondad siempre serán mi guía y ejemplo a seguir.

A mi familia paternal y maternal, gracias por todo el apoyo brindado y sus manifestaciones de cariño desde la distancia, por sus consejos e historias que me han servido de inspiración y guía. A mis primos, quienes, a pesar de los años, siempre me hacen sentir como un niño de nuevo y logran hacerme olvidar cómo se sentiría el hecho de tener un hermano. También, quiero agradecer a mi padrino, que aunque ya no estas físicamente presente con nosotros, junto a mi madrina fueron un apoyo incondicional e invaluable en mi vida, un ejemplo de perseverancia, disciplina, coraje y amor que forjaron en gran medida la persona que soy hoy en dia.

A mi compañera de vida, gracias por tu apoyo incondicional en cada momento y lugar. Sin importar el destino, siempre estabas presente con tu paciencia, amor y locuras. A mis amigos en Colombia, México y Canadá que a pesar de la distancia me dan su voz de aliento y están presentes siempre en los buenos y malos momentos.

Al Prof. Jesús Carretero Perez por darme la oportunidad y voto de confianza para realizar mis estudios de doctorado. Gracias por la paciencia, dirección y conocimientos brindados durante esta etapa de mi vida. Al igual que al Dr. José Luis González Compeán por su voto de confianza, asesoramiento y gran apoyo durante mi formación académica.

Al grupo de investigación ARCOS y compañeros de trabajo por su paciencia, enseñanzas y conocimientos brindados. En especial, a mis compañeros Diego, Eduardo, Dante y los Doctores Felix y Alex que me apoyaron en momentos difíciles de mis estudios y que hicieron parte de la realización de este trabajo de tesis. A los Doctores Tom y Orcun, por la oportunidad brindada y apoyo recibido durante mi estancia de investigación. También a ustedes, gran parte de este trabajo no hubiera sido posible.

A cada uno de ustedes y los que he pasado por alto, gracias por su apoyo.

This work has been partially supported by the project "CABAHLA-CM: Convergencia Big data-Hpc: de los sensores a las Aplicaciones" S2018/TCS-4423 from Madrid Regional Government.

Published and Submitted Content

During the process of researching and writing this dissertation, the results of this study have been presented and published in a variety of academic conferences and journals for review and evaluation by the academic community.

1. **Martinez-Rendon, C.**, González-Compeán, J. L., Sánchez-Gallegos, D. D., & Carretero, J. (2023). CD/CV: Blockchain-based schemes for continuous verifiability and traceability of IoT data for edge–fog–cloud. Information Processing & Management, 60(1), 103155. Q1.

   - DOI: `https://doi.org/10.1016/j.ipm.2022.103155`
   - URL: `https://www.sciencedirect.com/science/article/pii/S0306457322002564`
   - Note: The material contained in this article has been partially included in Chapter 3 of this thesis. The content of this item has not been singled out with typographical means and references throughout this thesis.

2. **Martinez-Rendon, C.**, Camarmas-Alonso, D., Carretero, J., & Gonzalez-Compean, J. L. (2022). On the continuous contract verification using blockchain and real-time data. Cluster Computing, 1-23. Q1.

   - DOI: `https://doi.org/10.1016/j.ipm.2022.103155`
   - URL: `https://link.springer.com/article/10.1007/s10586-021-03252-0`

- Note: The material contained in this article has been partially included in Chapter 4 of this thesis. The content of this item has not been singled out with typographical means and references throughout this thesis.

3. Carretero, J. and **Martinez-Rendon, C.**: Blockchain-based schemes for continuous verifiability and traceability of IoT data. Workshop BDCSA2023: Big Data Convergence: from Sensors to Applications. 31st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP 2023). March 1-3, Naples, Italy.

4. **Martinez-Rendon, C.**, Sánchez-Gallegos, D., Carretero, J., González-Compeán, J. L., Rubio-Montero, A. J. & Asorey, H. Calibración y evaluación experimental del Gestor CD/CV. Workshop CABAHLA. Jornadas Sarteco 2022. Alicante. 21 al 23 de septiembre de 2022.

5. Carretero, J.,**Martinez-Rendon, C.**, González-Compeán, J. L., & Sánchez-Gallegos, D. D. CD/CV: Blockchain-based schemes for continuous verifiability and traceability of IoT data for edge-fog-cloud. Information Processing & Management Conference 2022. 20-21 October 2022. Wuhan. China. Online presentation.

Other research merits

Throughout the period of my research on this dissertation, I had the opportunity to collaborate with other authors on various publications, which are not incorporated in this present document.

1. Yang, D., **Martinez, C.**, Visuña, L., Khandhar, H., Bhatt, C., & Carretero, J. (2021). Detection and analysis of COVID-19 in medical images using deep learning techniques. Scientific Reports, 11(1), 19638. Q1.
   - DOI: `https://doi.org/10.1038/s41598-021-99015-3`
   - URL: `https://www.nature.com/articles/s41598-021-99015-3`
2. Lopez-Arevalo, I., Gonzalez-Compean, J. L., Hinojosa-Tijerina, M., **Martinez-Rendon, C.**, Montella, R., & Martinez-Rodriguez, J. L. (2021). A wot-based method for creating digital sentinel twins of iot devices. Sensors, 21(16), 5531. Q1.
   - DOI: `https://doi.org/10.3390/s21165531`
   - URL: `https://www.mdpi.com/1424-8220/21/16/5531`

Research Stay

1. Argonne National Laboratory - Lemont, IL, USA
   - Supervisor: Tom Peterka and Orçun Yildiz
   - Research group: Mathematics and Computer Science Division (MCS)

- Research project: Triple Convergence of HPC, BD, and AI through ASCR In Situ Workflow Tools
- Period: 07/08/2022 - 07/11/2022

UNIVERSITY CARLOS III OF MADRID

Abstract

School of Engineering

Computer Science and Engineering Department

Ph.D. in Computer Science and Technology

Title

by Cristhian MARTÍNEZ RENDÓN

Nowadays, regardless of the use case, most IoT data is processed using workflows that are executed on different infrastructures (edge-fog-cloud), which produces dataflows from the IoT through the edge to the fog/cloud. In many cases, they also involve several actors (organizations and users), which poses a challenge for organizations to establish verification of the transactions performed by the participants in the dataflows built by the workflow engines and pipeline frameworks. It is essential for organizations, not only to verify that the execution of applications is performed in the strict sequence previously established in a DAG by authenticated participants, but also to verify that the incoming and outgoing IoT data of each stage of a workflow/pipeline have not been altered by third parties or by the users associated to the organizations participating in a workflow/pipeline. Blockchain technology and its mechanism for recording immutable transactions in a distributed and decentralized manner, characterize it as an ideal technology to support the aforementioned challenges and challenges since it allows the verification of the records generated in a secure manner. However, the integration of blockchain technology with workflows for IoT data processing is not trivial considering that it is a challenge not to lose the generalization of workflows and/or pipeline engines, which must be modified to include the embedded blockchain module. The main objective of this doctoral research was to create new techniques to use blockchain in the Internet of Things (IoT). Thus, we defined the main goal of this thesis is **to develop new techniques to integrate blockchain in Internet of Things scenarios for massive data management in edge-fog-cloud environments.** To fulfill this general objective, we have designed a content delivery model for processing big IoT data in Edge-Fog-Cloud computing by using micro/nanoservice composition, a continuous verification model based on blockchain to register significant events from the continuous delivery model, selecting techniques to integrate blockchain in quasi-real systems that allow ensuring traceability and non-repudiation of data obtained from devices and sensors. The evaluation proposed has been thoroughly evaluated, showing its feasibility and good performance.

UNIVERSIDAD CARLOS III DE MADRID

Resumen

Escuela Politécnica Superior

Departamento de Informática

Doctorado en Ciencia y Tecnología Informática

Titulo

por Cristhian Martínez Rendón

Hoy en día, independientemente del caso de uso, la mayoría de los datos de IoT se procesan utilizando flujos de trabajo que se ejecutan en diferentes infraestructuras (edge-fog-cloud) desde IoT a través del edge hasta la fog/cloud. En muchos casos, también involucran a varios actores (organizaciones y usuarios), lo que plantea un desafío para las organizaciones a la hora de verificar las transacciones realizadas por los participantes en los flujos de datos. Es fundamental para las organizaciones, no solo para verificar que la ejecución de aplicaciones se realiza en la secuencia previamente establecida en un DAG y por participantes autenticados, sino también para verificar que los datos IoT entrantes y salientes de cada etapa de un flujo de trabajo no han sido alterados por terceros o por usuarios asociados a las organizaciones que participan en el mismo. La tecnología Blockchain, gracias a su mecanismo para registrar transacciones de manera distribuida y descentralizada, es un tecnología ideal para soportar los retos y desafíos antes mencionados ya que permite la verificación de los registros generados de manera segura. Sin embargo, la integración de la tecnología blockchain con flujos de trabajo para IoT no es baladí considerando que es un desafío proporcionar el rendimiento necesario sin perder la generalización de los motores de flujos de trabajo, que deben ser modificados para incluir el módulo blockchain integrado. El objetivo principal de esta investigación doctoral es **desarrollar nuevas técnicas para integrar blockchain en Internet de las Cosas (IoT) para la gestión masiva de datos en un entorno edge-fog-cloud**. Para cumplir con este objetivo general, se ha diseñado un modelo de flujos para procesar grandes datos de IoT en computación Edge-Fog-Cloud mediante el uso de la composición de micro/nanoservicio, un modelo de verificación continua basado en blockchain para registrar eventos significativos de la modelo de entrega continua de datos, seleccionando técnicas para integrar blockchain en sistemas cuasi-reales que permiten asegurar la trazabilidad y el no repudio de datos obtenidos de dispositivos y sensores, La evaluación propuesta ha sido minuciosamente evaluada, mostrando su factibilidad y buen rendimiento.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

API	Application Programming Interface
BC	Blockchain
CD	Continuous Delivery
CV	Continuous Verification
DAG	Directed Acyclic Graph
DIY	Do It Yourself
ETL	Extract-Transform-Load
GM	Global Manager
HDF5	Hierarchical Data Format version 5
IoT	Internet of Things
MPI	Message Passing Interface

CHAPTER 1

INTRODUCTION

This chapter discusses the research problem of the thesis, presenting both the motivation and the definition of the research problem. In addition, the objectives and methodology used for the research work presented in this document are described.

1.1 Motivation

Internet of Things (IoT) is a paradigm in which physical objects, with features such as computing, communication, and sensing capabilities, are deployed in the field. These sensors and IoT devices collect and transmit environmental variables to the fog/cloud in real time and on a large scale, which represents a great challenge for the storage, processing and security in the life cycle of this data considering the volume of data that can be generated. According to a report by the International Data Corporation (IDC), the volume of data produced by 41.6 billion devices is estimated to be 79.4 zettabytes (ZB) of data by 2025. This growth in IoT data is largely due to the fact that IoT is useful in a wide variety of applications and sectors.

Some of the most common use cases for IoT include industry and manufacturing where it monitors and optimizes the production and supply chain as well as predicting and preventing machine problems and improving energy efficiency. In the health and wellness sector, it is widely used to monitor and track patient health, improve efficiency and safety in hospitals. In the field or agriculture, they are used to monitor and optimize the use of resources, such as water and fertilizers, efficiency in crop production or prevent possible fires. Sectors such as the home and smart cities also incorporate the use of IoT data to control, monitor and optimize resources. These are just some of the most common IoT use cases, but the technology is also being used in other areas, such as transportation, logistics and entertainment, among others.

Nowadays, regardless of the use case, most IoT data is processed using workflows that are executed on different infrastructures (edge-fog-cloud), which produces dataflows from the IoT through the edge to the fog/cloud. In many cases, they also involve several actors (organizations and users), which poses a challenge for organizations to establish verification of the transactions performed by the participants in the dataflows built by the workflow engines and pipeline frameworks. It is essential for organizations, not only to verify that the execution of applications is performed in the strict sequence previously established in a *DAG* by authenticated participants, but also to verify that the incoming and outgoing IoT data of each stage of a workflow/pipeline have not been altered by third parties or by the users associated to the organizations participating in a workflow/pipeline. Those verification and validation tasks result crucial for organizations to carry out critical decision-making processes in a confident manner, as any alteration in any data exchange performed by the workflow/pipeline applications would produce erroneous information, which would hinder the work of decision-makers. For example, several authors, such as [1], have highlighted the importance of adopting IoT in logistics to reduce the operating costs, to improve the capacity to respond to the changing needs of both the environment and customers, and to facilitate decision-making. The key factor lies in the agility of data acquisition from the IoT environment in real-time, which allows for faster processing of the data as soon as it is available.

One approach to face up this issue is monitoring and registering all transactions performed by the applications, as well as the alterations of the contents performed during the transactions, in a trusted environment using public/private blockchain networks. Blockchain technology allows the creation of distributed, immutable and verifiable records between two or more parties without requiring a centralized intermediary. The information is stored in the form of chained blocks that are linked by a unique cryptographic hash and where each block in the chain contains a series of transactions. This hash allows easy identification of data manipulation since if a block-/transaction is altered, the following blocks in the chain are also altered, which makes it very difficult for a possible attacker to modify the stored information without being detected. The use of blockchain technology has been popularized mainly for its use in cryptocurrency, but it also has wider applications not only in Supply Chain traceability but also in Electronic Voting, Healthcare System, Digital Right Management, Insurance, Financial System, or Real Estate [2]. The interest is due to the characteristics of the technology, which provides multiple benefits such as the complete operation of the system without relying on intermediaries or centralized entities, decisions made by all participants in the network, system transparency, immutability, reliability, and decoupled verification and processing of the transactions.

In [3] and [4], the authors studied some cases using blockchain in supply chain management system. As a result they concluded that it was emphasized that enabling secured transactions without trusted third parties in an automatic way makes legal and regulatory decisions much simpler. One of the areas that could benefit most is the logistics of food supply chains. Authors, such as [5], describe how to use blockchain technology in food supply chains and mention the main benefits of applying it, among which are more intelligent decision-making both for the client, the producer and auditors. Moreover, the possibility of achieving inter-organizational trust is a major advantage towards the adoption of blockchain technology in supply chains management [6].

1.2 Problem statement

Currently, large volumes of IoT data are produced and need to be processed to extract knowledge from them in a timely manner for decision making. One mechanism widely used in the literature for this purpose is pipelines and workflows. These mechanisms encompass a set of heterogeneous services/applications that apply a certain process to data for information extraction. These processes can be applied independently by different geographically distributed and unrelated organizations or entities. This scenario generates the need to establish control mechanisms to ensure full compliance and correct execution of each of the processes applied to the data by the different organizations.

On the one hand, it is necessary to establish control over the sequence of processes. Depending on the use case, there will be scenarios where two or more processes can be applied simultaneously to a set of data and there will be cases where there is process dependency. In other words, the execution of process i of organization A is required before executing the following process (i+1) of organization B. In this sense, it is vital to ensure the execution of organization A's processes so that organization B can operate.

Additionally, in scenarios where there is process dependency and in general, there could be failures/alterations in the execution of processes that critically affect the decision making process. If a participant in the workflow decides to act maliciously for personal gain or other purposes outside of what has been agreed upon by all parties involved, a mechanism must be in place to identify it. Therefore, it is essential to identify possible omissions or alterations produced by the organizations or entities involved in the processing of the data.

Blockchain technology and its mechanism for recording immutable transactions in a distributed and decentralized manner, characterize it as an ideal technology to support the aforementioned challenges and challenges since it allows the verification of the records generated in a secure manner.

However, the integration of blockchain technology with workflows for IoT data processing is not trivial considering that it is a challenge not to lose the generalization of workflows and/or pipeline engines, which must be modified to include the embedded blockchain module.

On the other hand, by including the verification component to the traditional workflow processes, there are estimated costs (overhead) of recording transactions performed by applications through edge-fog-cloud environments, which produces an efficiency problem that manifests itself in the form of delays in the delivery of data to the applications that create the dataflows and in the delivery of information assets to the decision-making processes.

Additionally, to ensure the recording of transactions in the blockchain, this network must produce and manage different cryptographic artifacts such as private keys and certificates to properly secure and guarantee the participation of organizations.

In this context, Figure 1.1 structures the previously described problems in three sections: 1) Workflow process verification, 2) Efficient IoT data processing and 3) Information management. Current traditional workflow engines and pipeline frameworks often do not take into account services to address these issues.

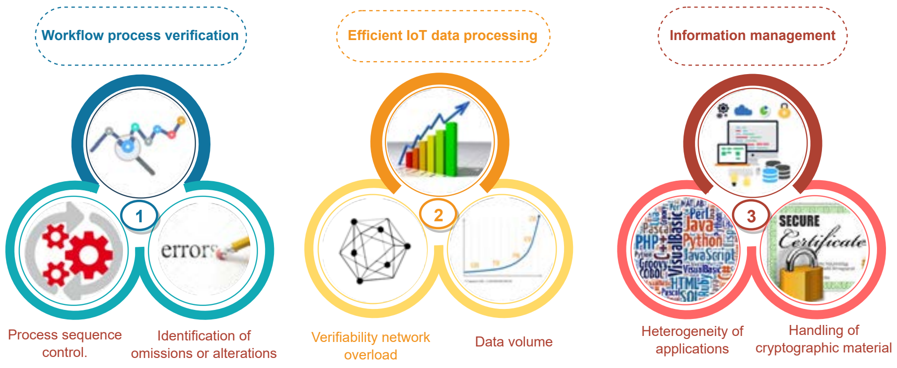

Workflow process verification
Efficient IoT data processing
Information management
1
2
3
errors
SECURE
Certificate
Process sequence control.
Identification of omissions or alterations
Verifiability network overload
Data volume
Heterogeneity of applications
Handling of cryptographic material

FIGURE 1.1: Problem statement

1.3 Hypothesis

The main hypothesis of this thesis is that the integration of blockchain techniques in IoT data management workflows could facilitate trust among actors and decision-making in almost real-time by making continuous verification of smart contracts applied to the different stages of IoT data processing. We foresee that platform-agnostic blockchain optimizations should provide better performance than that currently provided by blockchain platforms.

In this context, the present research work proposes the creation of new techniques to use blockchain in Internet of Things (IoT) domains and controlled workflows with execution commitments specified by users. These techniques focus on optimizing tools for use in near-real-time systems to ensure non-repudiation and traceability of data generated in workflows from IoT data obtained from devices and sensors. These aspects are crucial in order to maintain the accuracy of data and to reproduce the results obtained in scientific research and critical data analysis. Additionally, it permits the verification of the results achieved, the inspection of the data analysis process to address queries, and the pinpointing of possible areas for refinement. In this sense, the following general objective is proposed together with the specific objectives that allow the fulfillment of the same.

1.4 Objectives

The main objective of this doctoral research was to create new techniques to use blockchain in the Internet of Things (IoT). Thus, we defined the main goal of this thesis is **to design new techniques to integrate blockchain in Internet of Things scenarios for massive data management in edge-fog-cloud environments**. These techniques will focus on the optimization of tools for their use in near-real-time systems to ensure traceability and non-repudiation of data obtained from devices and sensors. In addition, the effects of these techniques for the implementation of controlled workflows with execution commitments specified by the users will be studied. The techniques provided should be infrastructure-agnostic to allow the usage of generic distributed processing structures to manage data life cycles, as well as to create one scheme for adding non-functional properties to the delivery of data.

To fulfill this general objective, the following specific goals are foreseen for this thesis.

1. To design a content delivery model for processing big IoT data in Edge-Fog-Cloud computing by using micro/nanoservice composition.
2. To design a continuous verification model based on blockchain to register significant events from the continuous delivery model, selecting techniques to integrate blockchain in quasi-real systems that allow ensuring traceability and non-repudiation of data obtained from devices and sensors.

3. To enhance the performance of blockchain systems to log IoT data to cope with the poor performance of current systems.
4. To evaluate the solution proposed with study cases to demonstrate its feasibility.

1.5 Contributions

The main contributions of this thesis are:

- Creation of a *Continuous Delivery/Continuous Verifiability* model allowing automatic deployment of integrated workflows for IoT-edge-fog-cloud systems and blockchain networks to record the actions performed in the workflows for continuous verification along system lifecycle. The deployment is made based on user specifications defined as extract/transform/load (ETL) operations and connections of components defined as a Directed Acyclic Graph (DAG). We have provided a technological solution for automatic deployment of a large scale distributed business logic system using virtualized appliances, including a usable Web interface to assist in deployment and visualization of data and results. We have also devised a formal method to describe the business logic and the different actors involved.
- Producing new techniques for reducing the overhead of transaction registration. Blockchain is slow to cope with real-time transactions, which affects the user service experience (decision-makers) of IoT workflows. We have enhanced the implementation of the blockchain transactions with two platform independent optimization techniques (atomic transactions and grouped validation). Moreover, we have enhanced transactions data transmission and recording by transparently providing parallel patterns for the blockchain process. Main

enhancements consist of performing real-time data acquisition for each sensor value independently and recording those values as single blocks in the chain. Those blocks are later consolidated in a bigger block for each sensor, but this is made outside the recording process and, thus, it does not delay data capturing.

- Automatic deployment of smart contracts. The installation and deployment of contracts in the CD/CV system is performed automatically using the ETL mechanism, which allows mapping each stage of the workflow with an organization in the blockchain. This mapping allows the verification of transactions in the different organizations or stages of the workflow, regardless of the specific purpose of each application. This generates an IoT data flow that maintains the generality of the solutions.
- A method for continuous verification of smart contracts in real-time. A reliable verification, involving trusted third parties in the loop, that allows very fast conflict resolution.

1.6 Research Methodology

The approach for carrying out the research project is depicted in Figure 1.2 and comprises of 5 primary phases: 1) Defining the scope, 2) Establishing the prototype, 3) Obtaining the necessary techniques, 4) Conducting experimental evaluation, and 5) Drawing conclusions.

Throughout these phases, an analysis of the current status of the subject matter will be undertaken to comprehend the overall theme and conceptualization of the issue. This understanding will be used to formulate, construct, implement and assess the techniques required for optimizing blockchain in IoT. These techniques will be designed to enable efficient and secure data

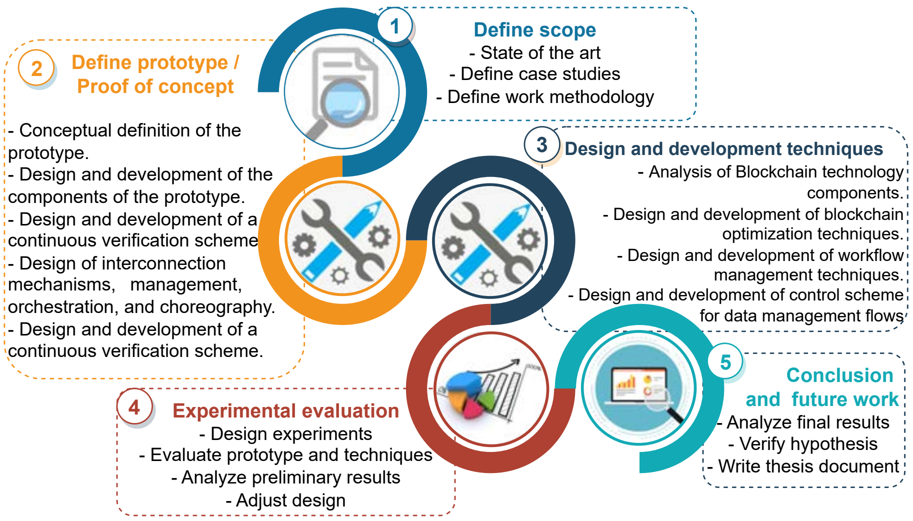

1
Define scope
- State of the art
- Define case studies
- Define work methodology
2
Define prototype / Proof of concept
- Conceptual definition of the prototype.
- Design and development of the components of the prototype.
- Design and development of a continuous verification scheme
- Design of interconnection mechanisms, management, orchestration, and choreography.
- Design and development of a continuous verification scheme.
3
Design and development techniques
- Analysis of Blockchain technology components.
- Design and development of blockchain optimization techniques.
- Design and development of workflow management techniques.
- Design and development of control scheme for data management flows
4
Experimental evaluation
- Design experiments
- Evaluate prototype and techniques
- Analyze preliminary results
- Adjust design
5
Conclusion and future work
- Analyze final results
- Verify hypothesis
- Write thesis document

FIGURE 1.2: Research Methodology

management through a workflow and control system, specifically for the handling of large amounts of data.

The forthcoming paragraphs outline the details of the four stages and the corresponding tasks that are anticipated to be executed in this research work:

1. **Define scope:** During this stage, 3 sub-steps were performed: 1) Definition of the state of the art, 2) analysis of possible case studies including IoT data availability and 3) definition of this research methodology.

2. **Define prototype/Proof of concept:** In this phase, we conducted a proof of concept where an initial prototype was devised and built to showcase the technical or functional feasibility before implementing the proposed optimization techniques. To achieve this, the subsequent tasks were performed: 1) The creation and construction of the

prototype components, 2) The formulation and execution of a continuous verification scheme, and 3) The design of interconnection, management, orchestration and choreography mechanisms.

3. **Design and development techniques:** The choice of optimization techniques was obtained by establishing the following sub-steps:

    - **Analysis of Blockchain technology components:** This analysis is focused on detecting possible Blockchain technology components that generate bottlenecks or directly affect system performance, especially in IoT scenarios where the amount of data to be processed is considerable.
    - **Design and development of blockchain optimization techniques:** The main goal of this phase was to conceptualize and establish the design principles of the optimization techniques. Throughout this stage, we identified the different components or elements of the proposed solution and adapted the approach to the design principles that are integral to Blockchain technology to ensure successful integration into IoT scenarios.
    - **Design and development of workflow management techniques:** During the progression of this phase, a model/schema for managing large-scale data processing was obtained. The subsequent tasks were undertaken to create this model: 1) Devising a data flow management scheme, 2) Creating and implementing techniques to optimize mass data processing using workflows and 3) Designing and constructing a workflow launcher that integrates Blockchain technology.
    - **Design and development of control scheme:** In this phase, the focus was on integrating the optimization techniques that were previously established to create the design for the Blockchain

network to be implemented. The primary purpose of this network is to regulate the management of IoT data flows. Furthermore, to ensure the effectiveness of the network, mechanisms for verifying transactions or processes carried out on the data were formulated. In addition, a traceability and non-repudiation viewer for workflows were conceptualized, constructed, and tested.

4. **Experimental evaluation:** The activities carried out in this stage include the integration in the base prototype of the Blockchain techniques optimized in two real scenarios, design of experiments, evaluation of the results obtained, analysis of the preliminary results and adjustments to the design of the initially proposed prototype.

5. **Conclusion and future work:** Finally, the present document was written to include the analysis of the final results and the verification of the hypothesis.

1.7 Dissertation outline

This document details the work conducted through the development of this thesis following the methodology described in the previous section, and it is structured as follows:

- Chapter 1, *Introduction*, has briefly presents the motivation, scope, and objectives of this thesis in the context of the problems currently evidenced.
- Chapter 2, *State of the Art*, establishes the basis of the contributions of this thesis, describing the current state of the main workflows and that of blockchain technology to address the described problem. This

chapter also compares the characteristics of the different solutions and presents the relevant limitations found in the literature.

- Chapter 3, *A methodology for continuous delivery/continuous verification schemes for traceable IoT dataflows*, presents the proposed Manager for integrating blockchain technology with IoT data workflows.

- Chapter 4, *Blockchain optimizations*, presents the optimizations obtained and applied to the proposed Manager.

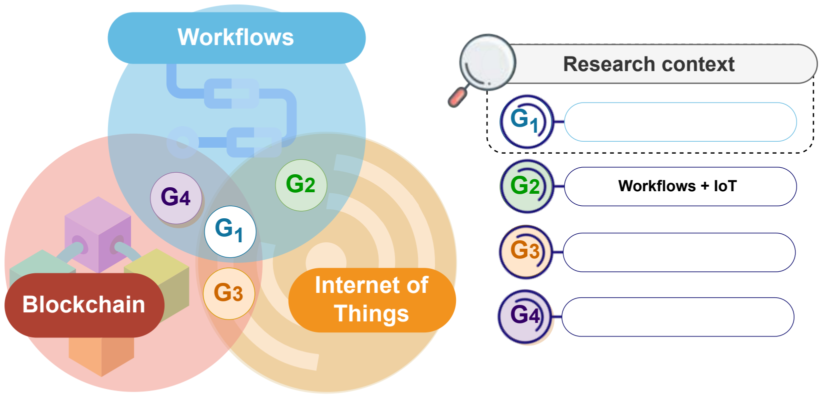
- Chapter 5, *Evaluation*, presents the results obtained by evaluating the proposed solution in different use cases.

- Chapter 6, *Conclusions*, summarizes this thesis and its objectives, detailing its contributions and results, while discussing potential directions for future research enabled by this work.

CHAPTER 2

BACKGROUND AND RELATED WORK

This chapter discusses the current status of various topics and areas related to the thesis. To provide a clear understanding of the related works, they have been grouped into three main sections. The first section covers workflow engines for IoT, the second focuses on data management in IoT workflows, and the third section is dedicated to Blockchain technology with a focus on describing its limitations and challenges. Finally, a related works section is presented to describe the works that cover some "convergence" between blockchain technology with workflows in the IoT environment, the proposed optimizations to the challenges of blockchain technology and the limitations found in those works. This organization allows a thorough and systematic exploration of the different aspects that are relevant to the thesis.



2.1 Background

Scientific workflow engines like Pegasus [7], Triana [8], and Sacbe [9] are primarily designed for handling heavy processes, and not for dealing with IoT environments. Although some solutions like DagOnStar [10] and Parsl (Scalable Parallel Scripting) [11] are based on libraries and can manage workflows consisting of Python functions and external applications on any computational resource, none of them are specifically designed for creating efficient workflows on IoT systems.

2.1.1 Workflows engines for IoT

New proposals have emerged to integrate IoT with workflows for real-time and automated decision-making environments. For example, a study [12] presented a proposal for using IoT in repetitive construction operations, while Osmotic flow computing [13] is a paradigm that allows for automatic deployment of microservices over interconnected Edge and Cloud Data Centers. In the Osmotic Flow model, an IoT workflow application is represented as a directed graph with data transformation tasks as its nodes and dataflow dependencies between them as its vertices. The FairWind system [14] is an intelligent navigation system that uses cloud computing and IoT to collect data from vessel sensors to create high-resolution 3-D maps of the ocean floor. However, FairWind is limited to a specific use case and depends on a third-party infrastructure like Amazon Web Services.

Recently, a new IoT workflow composition system (IoTWC), which allows IoT users to define their workflows with proposed IoT workflow activity abstract patterns (e.g., data capture, data store, data visualization), has been presented [15]. IoTWC leverages the analytic hierarchy process (AHP) to compose the multi-level IoT workflow that satisfies the requirements of any IoT application. An analysis, made using a smart home scenario, showed the effectiveness of IoTWC in terms of IoT workflow abstraction and composition. However, the paper lacks a real-world evaluation to measure performance of the solution, that seems to be mostly theoretical. All the former tools lack verifiability and traceability in the workflow stages deployed.

From the literature reviewed, we could see that major challenges for IoT workflows in collaborative edge, fog, and cloud environments are related to data placement strategies [16], [17], providing self-adapting services orchestration [18], interoperability of data and processes between the different stages [17, 19], portability of tasks [20], computation offloading [21], programmability and flexibility [22], efficiency [23], and finally verifiability.

Verifiability is a critical property in IoT workflows where data are used to control quality of goods, medicines, food, etc. [24]. However, a major challenge is that the data processing in IoT workflows can result in incorrect results, making them untrustworthy. Therefore, securely and reliably processing the outsourced data is a crucial security issue with many challenges. To address this problem, one proposed solution is to combine blockchain and smart contracts with IoT operations [25],[26],[27],[28],[29],[30]. For example, in [31], the authors demonstrated a blockchain-based supply chain traceability system for food safety, while in [32], they presented a blockchain tokenizer for industrial IoT applications that lack trust.

However, the unique characteristics of IoT systems, such as heterogeneity and pervasiveness, pose challenges in designing smart contracts for such systems [33],[34]. Moreover, even if the blockchain technology provide decentralized security and privacy [35], efficiency is a major issue, as it may involve power surge, high latency operations, and considerable computation, which is not adequate in some settings [36]. A cost analysis of internet of things sensor data storage on blockchain via smart contracts is shown in

[37]. The study showed that even though expensive, for those applications where the integrity and transparency of data are crucial, storing IoT sensor data on Ethereum could be a reliable solution.

2.1.2 Data management in IoT workflows

A comprehensive and systematic review of the existing literature on big data management techniques in the Internet of Things (IoT) is presented in [38]. The study explores the various big data management techniques that have been proposed and applied in the context of the IoT, including techniques for storage (e.g. distributed databases, cloud, in-memory or distributed file system), processing (real-time, batch, cloud or distributed), analysis (e.g. data mining, machine learning, sentiment analysis, social networking) and data visualization (e.g. real-time, static or geospatial data).

From a more generalized and reduced point of view, the typical data handling that is done in workflows is described in Figure 2.1.

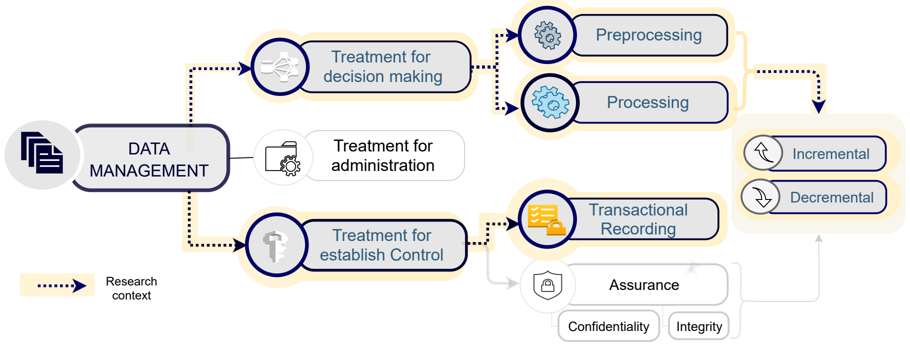


FIGURE 2.1: Taxonomy of techniques of data management in IoT.

The taxonomy that is illustrated in the figure is based on the work presented in [39]. In this taxonomy, there are three types of data processing:

- Treatment for decision-making. It basically consists of the processes applied to the data either for the extraction of knowledge and patterns in the data (decremental process), or to generate added value to them (incremental process) [40].
- Administration of meta-data. It refers to the management of the meta-data of the data, such as where the data is located, where it comes from, who it is, size, etc. [41].
- Procedures to establish control of the data. It basically corresponds to two tasks, the first is the traceability record of the operations on the data necessary to avoid problems such as non-repudiation [42], and the second task is the assurance (for example, confidentiality and integrity) that is required in sensitive data now. They are transmitted during the different stages of the workflow and primarily in cloud environments [43].

In addition, in [44] the authors present an insight into the data management techniques for IoT. The study includes the most relevant concepts of IoT data management and a comparison of surveys on the topic. The authors highlight new IoT data management methods, such as middleware or architecture-oriented solutions to facilitate efficient storage, the integration of generated data, and indexing methods of structured and unstructured data. In general, the authors classify these methods around three main principles: data collection, data management system design, and data processing. As future work, the authors propose mechanisms using future technologies such as Fog computing to improve data management.

Transactional Recording

The traditional solutions for verification of transactions are focused on a single piece of software with which the end-users are dealing with. This type of solution is not suitable for workflows used in IoT dataflows, as a workflow is composed by different pieces of software (applications).

Security is a critical aspect of managing IoT data. As a result, various studies have suggested integrating blockchain solutions into IoT systems to enhance security in data access, transmission, and storage [45, 46]. Blockchain-based solutions provide trust and stability, making them a suitable option [47]. Recently, Puri et al. [48] have proposed using smart contract-based policies to reinforce privacy and security in IoT.

Some solutions have been proposed to avoid Over-centralization of data. Ramachandran et al. [49] describe different approaches for distributing the verification and data management aspect between Linked Data and the distributed ledger, keeping the integrity of the stored information intact through Blockchain-based verification. Helo presented in [50] a pilot system of a cloud-based portal for real-time tracking and tracing of logistics and supply chains using IoT and blockchain. The architecture of the proposed portal system is connected to transport companies, tracking devices, consolidation points and suppliers.

Braun et al. [51] have recently presented an approach for documenting the execution of inter-organizational workflows on a distributed ledger, with the possibility of adding selectively shared verifiable data to the workflow instances' documentation. It allows to record events, but the solutions is tailored to the specific use case. Shukla et al. [52] have presented this year a system called BlockIoT which connects personal Internet of Medical Things (IoMT) devices to Electronic Health Records using blockchain technology to provide a trustworthy and reliable method for aggregating IoMT device data and smart contracts that automatically provide relevant alerts to the healthcare providers.

2.1.3 Blockchain technology

Classification

Blockchain technology can be classified attending to the nature of the blockchain network created. Different studies analyze the different development tools and network taxonomies available to determine which one provides better security, confidentiality, immutability, decentralization, anonymity, or auditability of data. Basically, there are three main types: public, private, and consortium. The different network taxonomies available in blockchain technology are compared in [53] and [54].

In public networks, anyone can participate and register operations in the blockchain. The transactions are visible to all participants and allow a completely decentralized blockchain network to be established. In addition, the high number of individuals connected in the network ensures that the stored data is immutable, since it is replicated by each of the participants, making it practically impossible to alter it. However, this information is visible to all members, which means that privacy is reduced, because anyone can perform and validate transactions without needing to belong to a specific organization. The Ethereum [55] and Bitcoin [56] development platform uses this type of network. A example of application is the platform called BanQu, which aims to connect the unbanked to the global economy through mobile phones and a blockchain network [57], those platform allows people to record their financial transactions in the blockchain history owning their data. The main characteristics of the former platforms are profitability, transparency using secure and auditable records, implementation anywhere, and sustainability.

Nevertheless, there are several issues associated with public and open networks. For instance, research conducted on BanQu [58] revealed that the storage location of the decentralized ledger is uncertain. In addition, it is challenging to deploy the technology infrastructure in the targeted customers' region. Another study [59] demonstrated that Bitcoin transactions can be linked to users, which jeopardizes anonymity. Moreover, the scalability of these solutions is typically poor since there is no limit to block size. To address this challenge, a novel protocol that focuses on scalability has been proposed in [60].

In private networks, only some peers have the capacity to record and manage operations in the blockchain system, being Hyperledger Fabric [61] the de-facto standard platform for this kind of networks. In contrast to public networks, private networks only have participants who are members of a particular organization [62]. This removes the anonymity of the participants in the transactions, because they must authenticate their identity with a certificate. In those networks, it is a lot easier to modify a transaction, but the immutability level is still high.

The third category of network taxonomy is consortium networks [63], which are a fusion of the two previous network types. These networks have public data, but only a particular group of participants are responsible for validating transactions. Hyperledger Fabric is an example of a development tool that can utilize this type of network. Currently, consortium blockchain and smart contract technologies are being used to secure data storage and sharing in IoT systems. For instance, the application of these networks in vehicular edge networks is demonstrated in [64].

Lastly, it is noteworthy to mention the investigations conducted by [65], [66], and [67], where they compare various development platforms, such as Hyperledger Fabric and Ethereum. They mainly focused on the dependability of the data stored in the transactions and the implemented network type. As highlighted in those studies, private networks provide better reliability since only trusted actors within the organization participate in the validation process. This feature also enhances security since participants outside the organization cannot view the content of the transactions.

In this research work, we have chosen to use Hyperledger Fabric, since its features, which are described in the following section, allow the creation of networks of entities with different privileges, ideal for integration with workflows where multiple actors are involved.

Hyperledger Fabric

Hyperledger Fabric is an open source blockchain technology toolkit created by the Linux Foundation, which aims to provide a flexible and scalable infrastructure for building blockchain-based enterprise applications [61].

Hyperledger Fabric achieves consensus by relying on a backend service (known as the ordering service) that intermediates the messages between senders and receivers [68]. This backend service will ensure that all receivers will see messages in the same order – it follows that if all receivers see messages in the same order, they will perform the same actions/commits, etc. and the consensus is achieved. The consensus method used in Hyperledger Fabric is called "Consensus Pluggable", which gives users the option to select different consensus algorithms according to their business needs and requirements.

Consensus algorithms available in Hyperledger Fabric include Practical Byzantine Fault Tolerance (PBFT), Raft and others. Currently Fabric 2.x uses Raft, a Leader/Follower type, consensus algorithm [69]. Moreover, Hyperledger Fabric uses Crash Fault Tolerance (CFT) to achieve consensus for single as well as multiple org systems. Crash Fault Tolerant model guaranties to withstand system failures, such as crashes, network partitioning. Having N nodes in your consensus system CFT capable to withstand up to N/2 such crashes.

During the consensus process in Hyperledger Fabric, network nodes (also called "peers") validate and confirm transactions, reaching consensus on the current state of the distributed ledger (known as the "ledger"). Each node in the network maintains a copy of the ledger, ensuring that all participants have access to the same information and that data integrity is maintained.

Hyperledger Fabric uses a modular network architecture to achieve consensus, which means that business logic and data are separated into different layers. This separation allows the creation of private channels and the establishment of access policies that control who can see what data and under what conditions. In addition, Hyperledger Fabric provides an identity and access management system that enables enterprises to control and manage permissions for users and stakeholders on the network.

In summary, Hyperledger Fabric uses a scalable and modular approach to consensus, giving users the ability to choose between different consensus algorithms and control access to data. This makes it a standout choice for enterprise blockchain applications that need a high level of security, privacy and flexibility.

Applications

One of the prominent uses of blockchain technology is in the field of cryptocurrency, such as Bitcoin, to enable financial transactions and supply chain financing [70, 71]. Some of the popular blockchain applications in this area include the utilization of cryptocurrencies to manage logistics and transport operations, as seen in the Chinese "One Belt One Road-OBOR" initiative [72]. This project aims to connect China with multiple countries across Eurasia, Africa, and Oceania. According to experts, the implementation of blockchain technology can significantly reduce the transfer time and costs of funds, making it a feasible financing option for OBOR projects.

In an effort to address the needs of the maritime logistics industry, [73] developed a blockchain-based token deposit system using Ethereum as the underlying technology platform. The system included a RESTful API for business integration and garnered the participation of several major carriers and shippers. However, the system faced two significant challenges which led to its suspension in October 2019. Firstly, the low volume of transactions made it difficult to continue commercial operations. Secondly, there were design issues, resulting in some shipments not being executed in accordance with reservations. Additionally, shippers faced difficulties in confirming their reservations during the high season due to performance problems.

In [5] and [74], studies are presented that explore the use of blockchain technology for provenance and traceability in the Internet of Things (IoT) integrated with food logistics. The authors describe different cases that have been implemented in various parts of the world. One of the cases studied involves Walmart using blockchain to monitor pork in China and mangoes in Mexico [75]. The results of the tests indicate that blockchain can significantly reduce the time required to track information, from one week down to just 2.2 seconds.

Various studies have implemented blockchain technology in supply chains due to its valuable benefits like traceability and trust, which enhance the dependability of supply chains. Although it can be applied to any product, blockchain is particularly useful for sensitive items such as food and medicine. For example, agri-food supply chains to certify and warranty the origin of goods to prevent frauds [5, 76–78]. Furthermore, it has been used in medicine supply chains, because many of them are sensitive to temperature or humidity, among other factors. Therefore, it is essential to ensure that the medicine is in optimal condition when it is consumed, following the Good Distribution Practice of medicinal products for human use (GDP).

For example, in [79] they use sensors to take measurements that are sent to a cell phone, which records the measurement in the ledger. Moreover, blockchain technology has been also used in supply chains for high-value items to prevent counterfeiting, such as in the case of Diamonds [80].

Bext360 [81] is a platform that aims to provide a traceable supply chain from producer to consumer. The platform offers configurable solutions and APIs that enable its technology to be integrated into various systems, such as supply chain management and point of sale systems. Data is collected at every stage of the supply chain and stored in an immutable record. However, according to [58], it is unclear how the data is handled in the later stages of the supply chain, which may result in the consumer not having a complete understanding of the supply chain.

In [82], guidelines for building a traceability system in the agrifood supply chain using RFID technology and blockchain are proposed. The authors suggest that RFID can be used to acquire and exchange data at different stages of the supply chain, while blockchain can ensure the reliability and authenticity of the shared information. However, the authors do not provide a prototype to demonstrate the potential benefits of their proposed solution. Instead, they focus on explaining where RFID tags should be placed, what information should be recorded, and how to use this information at each stage of the food supply chain.

In the study presented in [83], the authors used blockchain technology to improve the traceability and transparency of current recycling systems. Their theoretical model analyzed the costs associated with implementing a traceability system and proposed ways to reduce costs to make the system cost-effective. They focused on determining the optimal amount of waste that could be exchanged between supplying and consuming companies to maximize profit based on parameters such as the number of suppliers, consuming companies and the processing capacity of consuming companies.

To estimate blockchain implementation costs, the authors used several use cases provided by blockchain solution provider companies.

In [84], a smart logistics architecture using Ethereum Smart Contracts, microservices, automatic learning, and big data was proposed. The solution includes a logistics planner and a contract supervisor for asset management. However, the study's scenario was deemed unrealistic due to its small scale (e.g., a single consumer, an asset) and lack of cost analysis for blockchain-based traceability. Additionally, the authors did not provide measures such as throughput or latency to indicate the system's data processing capabilities.

However, as explained in [85], [86] and [4], the majority of works focuses on studying and describing conceptual or theoretical models of potential applications of this technology in supply chains, but they do not explain the technical details to implement the model or analyze its performance to determine if it is feasible to use blockchain in production of the described use case.

2.2 Related Work

This section outlines the studies conducted within the state of the art that have the most relevance to the research work at hand.

2.2.1 Optimizations in Blockchain platforms

Blockchain performance has been always an issue, especially when applied to real-time systems, like IoT. Several authors have made performance analysis on the different available development platforms [87, 88] and other studies have recently proposed optimizations to increase the performance and scalability of the tools based on RAFT [89, 90]. Those aspects are especially important for those environments including IoT devices, which send a big volume of data in a short time, making the management and storage of the data more difficult. However, most works propose to modify the architecture of the Fabric development tool so that the validation phase has a shorter execution time, as it is the main bottleneck detected in blockchain technology. The result is that those optimizations depend on the development tool and the modified consensus protocol used to validate the transactions. Thus, they are not generic optimizations and in many cases they are not adopted by the users.

Since blockchain consensus protocols are very complex, several publications, pursuing to increase blockchain performance, have carried out a thorough study to determine the bottlenecks that exist when a new transaction is processed and stored into the database [91–93]. Particularly in [92], the study describes the performance and optimization of the Hyperledger Fabric blockchain platform. In this study, extensive testing and analysis of different blockchain network configurations are conducted to determine the impact on the performance and efficiency of the platform. In addition, the authors propose and evaluate several strategies to improve the performance and scalability of the network, such as block size optimization, implementation of resource isolation policies, and equal distribution of nodes. The main objective of the study is to provide a practical guide for blockchain developers and architects working with the Hyperledger Fabric platform to improve the performance and efficiency of their blockchain applications.

Following the bottlenecks detected, different optimizations have been proposed to reduce the latency in transaction processing and storage. Those optimizations mainly consist in the implementation of local caches, parallelization of the transaction validation, or massive and parallel writing and reading in the database that stores the world state [94–98]. Other interesting improvements, try to reduce the size of the messages by separating the transaction header from the body, which contains the relevant content of the transaction, to send only the transaction identifier and the body, thus reducing the size of the transaction [97]. Finally, another possible optimization is to use a hash table in memory to store the world status, thus increasing performance as access is made to memory and not to hard disk [99].

In addition to these optimizations that affects directly to the transaction validation phase, there are other optimizations that are applied to the blockchain network configuration to increase the number of transactions that can be executed. One possible improvement proposed [100] consists of dividing the peer nodes into two different roles: Commitment and Endorsement, allowing the peer nodes to require fewer resources and to be able to scale the network on demand. Other optimizations consist of adjusting parameters, such as the number of channels on the network, the database used to store the world state, or the resources available to the nodes (e.g. number of CPUs or the network bandwidth) among other parameters. Thakkar [92] made a performance analysis using this type of optimizations and noticed that the number of transactions increased with some of the proposed configurations, thus increasing the performance.

As shown above, most works propose to modify the architecture of the development tool so that the validation phase has a shorter execution time, as it is the main bottleneck detected in blockchain technology [92, 94, 97]. However, these optimizations depend on the development tool and the consensus protocol used to validate the transactions. Thus, they are not generic optimizations.

The performance limitations are evident in terms of transaction volume, latency, and block size [53]. On the other hand, most proposals consider a predefined scenario or a time-invariant data set. Usually, a monolithic deployment is performed where the verifiability network and the proposed solution are on the same infrastructure. In many cases, these deployments are performed on virtual machines and require the knowledge of an expert, limiting their scalability and their performance (in contrast to containers [101, 102]). Additionally, communication between different blockchain implementations is not possible at this time, preventing interoperability and widespread adoption for data storage and analysis [103]. Finally, in the validation of accumulated data that occurs in the final stages of the value chain, they limit or prevent the use of early corrective actions to reduce the cost of damage caused.

2.3 Summary

Table 2.1 shows a summary of similar solutions from the state-of-the-art focused on the management, processing, and traceability of data through the edge-fog-cloud. We have classified these solutions according to:

1. the scope of the solution (Blockchain: use smart contracts and Workflow: structures for the processing of data),
2. their environment (Multiple: if the solution is deployable in any of the edge, the fog, and the cloud),
3. their deployment (Distributed: on multiple compute nodes and Container: encapsulated and interoperable solution),
4. the data verification process (Real-Time: Receiving real-time data streaming, Continuous: issuing alerts, and traceability at any time of the data lifecycle),
5. whether they include a visualization component (e.g., Dashboard to visualize the resulting data or analysis),

6. the domain or use case (Multiple: solution available for different use cases, and IoT: If the data flow is coming from IoT devices),
7. the use of efficiency techniques to reduce the time required to manage and process the data (Parallelism: if data processing or verification processes in parallel and Scalable: when the solution allows increasing its nodes/services without ceasing to operate).

	Scope		**Environment**	**Deployment**		**Verification**		**Visualization**	**Domains**		**Efficiency**	
Work	Blockchain	Workflows	Multiple	Distributed	Container	Real-Time	Continuous	Dashboard	Multiple	IoT	Parallelism	Scalable
Jenkins [104]	-	✓	-	✓	✓	-	-	-	✓	-	✓	✓
Pegasus [105]	-	✓	-	✓	✓	-	-	-	✓	-	✓	✓†
Slurm [106]	-	✓	-	✓	-	-	-	-	✓	-	✓	✓
PuzzleMesh [107]	-	✓	✓	✓	✓	-	-	-	✓	-	✓	✓
DagOnStar [10]	-	✓	✓	✓	✓	-	-	-	✓	✓	✓	✓
Parsl [11]	-	✓	-	✓	✓	-	-	-	✓	-	✓	✓†
Martinez [108]	✓	✓	-	✓	✓	✓	✓	✓	-	✓	-	✓
Sacbe [9]	-	✓	-	✓	✓	-	-	-	-	-	✓	-
Nasir [88]	✓	-	-	-	✓	✓	-	-	-	-	-	✓
Ramachandran [49]	✓	-	-	✓	✓	✓	-	✓	-	-	-	-
Helo [50]	✓	-	-	✓	✓	✓	✓	✓	-	✓	-	✓
Braun [51]	✓	✓	-	✓	-	-	-	-	✓	-	-	-
Shukla [52]	✓	✓	-	✓	-	✓	✓	✓	-	✓	-	✓
CD/CV	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

† means that the characteristic is provided by external tools.

TABLE 2.1: Summary of state of the art - General solutions.

Solutions such as Jenkins [104], Pegasus [105], Slurm [106], PuzzleMesh [107], DagOnStar [10], Parsl [11] among others, allow the construction of workflows for multiple use cases including IoT data management. Additionally, these solutions are intended to be distributed and deployed in different infrastructures (e.g., in any of the edge, the fog, or the cloud), including parallelism patterns that allow them to be efficient and scalable. However, they lack the verification component to ensure the traceability of data or the correct execution of the processes that are part of these workflows. Ramachandran in [49] include the blockchain for a complete decentralized verification of data with confidentiality, however, such work is a conceptual solution that is in the process of development and testing. Implemented works such as that of Helo [50] and Shukla [52] use the blockchain for process verification on IoT and real-time data but are designed and defined for a particular use case, the former for logistics/supply chains and the latter for Electronic Health Records. The latter, together with Braun's work [51], integrate blockchain technology with a workflow in a single solution. However, Braun's work does not indicate its applicability in an IoT data environment but focuses on the execution of an interorganizational workflow using the distributed ledger.

2.3.1 Limitations of Previous Work

A summary of the features of the solutions analyzed with blockchain is presented in Table 2.2. Their main strengths and limitations using major features are discussed below.

1. **Generality and performance:** We have identified the main performance bottlenecks in blockchain and the solutions that have been proposed in the literature to solve them to obtain better performance and scalability of the Blockchain platform. The performance limitations are evident in terms of transaction volume, latency, and block size, or in bytes of transactions related to traditional systems such as Visa or Mastercard [53] where about 50,000 t/s are processed. This demand for transactions from traditional applications has led to strong efforts to optimize each development platform or a specific consensus protocol. However, this approach prevents a generic coupling of the solutions proposed to other blockchain platforms.

2. **Blockchain-IoT integration:** The design of most of the works analyzed does not allow the integration of blockchain technology with systems that support the data load produced by different IoT devices.

The majority of the proposals consider a predefined scenario or an already established and unchanging data set over time.

3. **Monolithic deployment:** A major limitation of the proposal found is the lack of dynamic deployment of the blockchain network together with the solution they propose. In many studies, the deployment of verification networks and the proposed solution are in the same infrastructure, so that it is not possible to have a decoupled, distributed, and decentralized system. In this sense, current work does not consider factors such as latency resulting from the geographical distribution of network nodes or scenarios in which possible network nodes or solution components fail.

4. **Manual and supervised deployment:** Part of the studies analyzed deploy the blockchain network manually through the knowledge of an expert in the area. This dependence on the expert's knowledge limits the scalability of the system in scenarios with a complex network, considering additionally that companies or entities not an expert in the use of blockchain could not deploy a solution for lack of technical knowledge in the area.

5. **Use of virtual machines:** With the advance of technology, different studies have highlighted the advantages and benefits obtained by the containers against the use of virtual machines [101, 102].

6. **Interoperability and integration with existing systems:** There are currently different implementations of blockchain designed by several companies. Communication between these implementations is not possible at this time, which prevents their interoperability and widespread adoption for data storage and analysis of higher-order and capacity [103].

7. **Post-registration verification:** Some works consider the registration of transactions in real-time, and other accumulate data before performing the registration. However, the validation of these accumulated data occurs in the final stages of the value chain, limiting or preventing the use of early corrective measures to reduce the cost of damage caused.

This research aims to address the problems discovered in the literature reviewed by designing, building, and deploying a system that includes all the fields analyzed in Table 2.2.

TABLE 2.2: Summary of state of the art - Blockchain solutions.

	Tec[a]		**Dep**[b]		**Ver**[c]		**Vis**[d]
	BC	IoT	Di	Co	RT	Cn	Da
Baliga [109]	✓	-	✓	-	-	-	-
Nasir [88]	✓	-	-	-	-	-	-
Pournader [110]	✓	-	-	-	-	-	-
Saberi [111]	✓	-	-	-	-	-	-
Tian [92]	✓	-	-	-	-	-	-
Javaid [94]	✓	-	✓	-	-	-	-
Gorenflo [97]	✓	-	✓	-	-	-	-
Arumugam [84]	✓	-	-	-	✓	-	-
Treat [82]	✓	✓	-	-	-	-	-
Leung [73]	✓	-	-	-	-	-	✓
Gadnis [57]	✓	-	-	-	✓	-	✓
Jones [81]	✓	✓	-	-	-	-	✓
Kamath [75]	✓	✓	-	✓	✓	-	-
CD/CV	✓	✓	✓	✓	✓	✓	✓

[a] **Tec**hnology (BC: Blockchain, IoT); [b] **Dep**loyment (Di: Distributed, Co: Container); [c] **Ver**ification (RT: Real-Time, Cn: Continuous); [d] **Vis**ualization (Da: Dashboard).

Unlike previous examples, the **CD/CV** schemes proposed in this thesis introduce an intermediary layer between the workflows components and the blockchain, enabling the workflows to remain generalized and easily adaptable for organizations. This intermediary layer facilitates the registration of application actions and the execution sequence of applications, while also providing automatic deployment of parallel patterns and consolidation of transaction registers to improve the performance of interaction with workflows and blockchain. The CV components enable the creation of continuous verifiability for transaction registration in distributed structures, making it possible to implement verifiability networks based on blockchain in edge-fog-cloud environments. This solution is applicable to any IoT data management workflow without requiring modifications to the applications themselves. The workflow stages provide real-time information, while blockchain technology ensures a chain of immutable transactions. This solution is described in detail in the next chapter.

CHAPTER 3

A METHODOLOGY FOR CONTINUOUS DELIVERY/CONTINUOUS VERIFICATION SCHEMES FOR TRACEABLE IOT DATAFLOWS

In this chapter, a scheme is explained for tracking the flow of data in IoT. The proposed method involves combining blockchain technology with a data processing model that is frequently used in workflows, known as continuous delivery.

3.1 Introduction

This research work presents a new approach for the development of CD/CV (**C**ontinuous **D**elivery/**C**ontinuous **V**erifiability) systems that incorporate blockchain technology with workflow engines and/or pipeline builders.

Figure 3.1 shows an example of the CD/CV scheme proposed. It shows two layers running in parallel but integrated: a Continuous Delivery system managing the IoT dataflows, and Continuous Verification system that allows to register the transactions performed by each stage of a IoT dataflow and verify them on-line using smart contracts.

The CD layer is based on techniques such as continuous delivery technique [104, 112], ETL and DAG. Continuous delivery is used in software industry for organizations to split systems into small manageable software pieces; as a result, the pieces can be distributed to different teams that could deploy/execute them through multiple infrastructures and integrate them

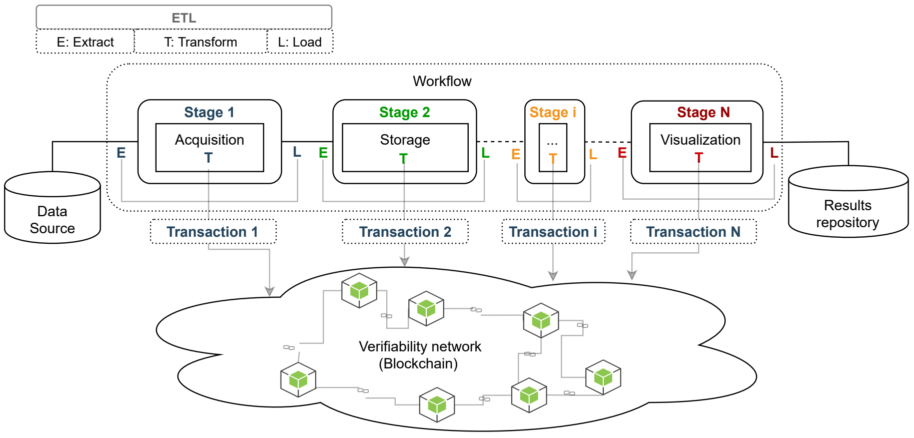


FIGURE 3.1: CD/CV model applied to a workflow.

into a single solution by using control structures based on directed acyclic graph (*DAG*). In this thesis, the coupling technique is reused to create edge-fog-cloud dataflows and to deploy the pieces on different infrastructures, as well as to invoke the components of the second part (CV for continuous verification) of the CD/CV schemes. The ETL processing technique (extract, transform, and load) [113] traditionally is used in big data scenarios for data analysis and the transfer of data between different databases. It is used in our CD model for allowing end-users to provide information about the path of the *E*xtraction of the input data and contents that will be used by the stages (i.e., data produced by an IoT device), the *T*ransformation performed by the applications associated to this stage to the extracted data/content, and the *L*oading of the transformed data/content into another stage or a storage location (e.g., a sink deployed on the cloud). In this structure, the nodes represent the applications of the stages whereas the edges represents the I/O interfaces.

The CV layer registers the transactions performed by each stage of a IoT dataflow (CD model), thus allowing end-users to create traceability reports and contract verification for each processing phase in an IoT dataflow. The

CD model automatically register the transactions performed by the applications by using ETL features and the *DAG* information to create trace routes of the information assets in the dataflow. The CV layer captures the operations performed by each software piece deployed on any of the edge, the fog, or the cloud infrastructures considered in an IoT dataflow. This produces a set of records in the blockchain network [114, 115], one per stage in a workflow, concatenated into a single register created by following the *DAG* of the CD model. That enables organizations and end-users to yield a continuous verification of the transactions performed through the workflows by using intelligent contracts on each stage.

3.2 Methodology for building CD/CV schemes

Figure 3.2 shows the methodology defined for building CD/CV schemes, consisting of three major steps: preparation, deployment, and operation.

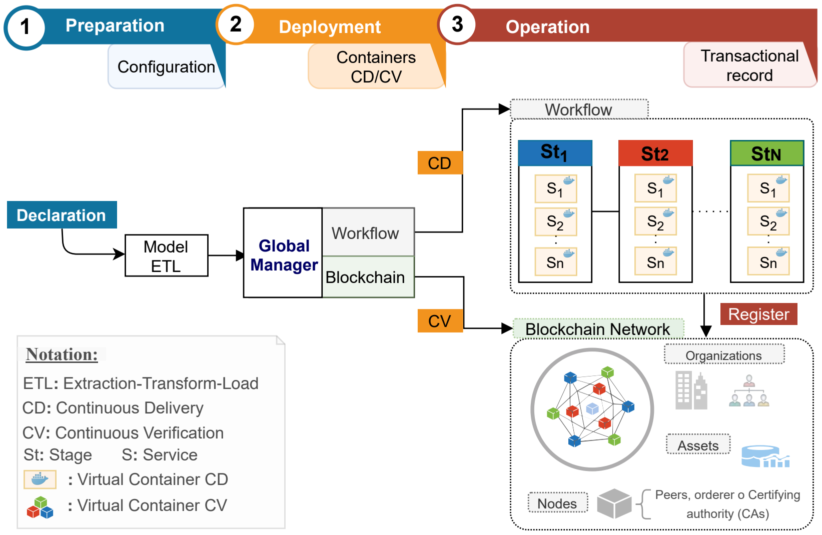


FIGURE 3.2: Methodology proposed for building CD/CV schemes.

The first phase is a *preparation* step in which all the stages (organizations) that want to participate in a IoT dataflow express ETL and DAG information.

In the *deployment* phase, a **Global Manager (GM)** receives the ETL and DAG information to ensure the effective materialization of this information in the form of traceable workflows by using the CD model. The (*DAG*) defining the dataflows is created to deploy them, components are inserted into the stages, using the ETL paradigm to extract data and events (transactions) from the CD components (stages, nodes, and edges) arising in IoT dataflows. The manager uses ETL components and the *DAG* to create and connect the different stages of the IoT dataflow, and it also establishes control structures for registering the transactions in the verifiability network. Finally, it checks that the continuous delivery (CD) and the continuous verification (CV) for the workflow are deployed. The CV model uses the information from the CD model to create two types of records in the blockchain: records of each service/app using the ETL information, and traceability records using the *DAG* scheme.

Finally, in the *operation* phase, software virtualization has been used to control the launching of both the workflow, and the verifiability network. Clones of the stages can be launched and managed as clusters in the form of parallel patterns (e.g., Manager/Worker pattern [116]), which improve stage performance and reduce the cost of the verifiability network.

3.2.1 Design of the CD/CV Global Manager

A manager is required to build and operate a CD/CV scheme. In this sense, Figure 3.3 shows the Global Manager proposed, which is composed of a *Construction Manager* and an *Operation Manager*.

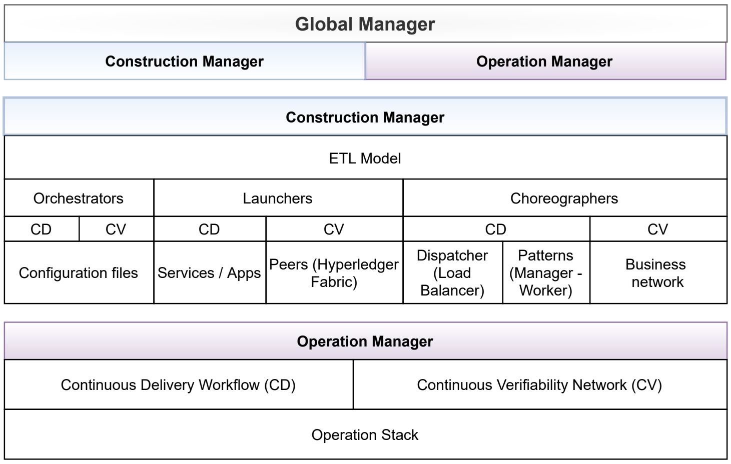


FIGURE 3.3: Modules of the Global CD/CV Manager.

The Global CD/CV Manager uses an ETL Interconnection model to get the information from the end-users to create a IoT dataflow and the verifiability network. In this sense, the ETL interconnection model together with the *DAG* scheme interconnect the workflow components with the verifiability network.

The CD components of the model that describe the workflow components are as follows:

1. **Data Source (DS):** It represents the source of data to be processed through a workflow (W).
2. **Service (S):** It represents the user application that performs specific processing (e.g., S_1: data analysis, S_2 : data compression, S_3 : encryption, among others.)
3. **Stage (St):** It represents the set denoted as $St = \{S_1, S_2, S_3, ..., S_m\}$, where m is the number of services (S) including in a given stage (St).
4. **Workflow (W):** The workflow describes the set of stages (St) that collaboratively perform general processing to transform the original data

source (DS) into useful information for entities or organizations that require to make decisions from the found information. A workflow is denoted as $W = \{St_1, St_2, ..., St_N\}$, where N is the number of processing stages, which are interconnecting ETL model and the *DAG* scheme.

5. **Extract, Transform, Load (ETL):** The ETL process allows a service (S_i) to obtain data from a given source (DS), applies given processing, and loads it in a specific storage system or space so that the result delivered by S_i is the input of the following service (S_{i+1}) to perform the same process, thus forming a workflow (W). This ETL process defined by each $S_i \in St \land St \in W$ consists of the following phases:

   (a) **Extract (E):** Indicates where to extract or acquire the data that will be the input of a S_i service. In this phase are considered two attributes $< type, path >$. Where $type$ indicates the input type that receives S_i: this can be a folder or a file, and $path$ indicating where are the data to be extracted.

   (b) **Transform (T):** Indicates how S_i (application/service) will process the data. In this phase, three attributes are considered: <$type$, $path$, $parameters$>. Where $type$ is the method through which S_i implements the data transformation. It can be a service or an executable (e.g., a JAR, python script, etc). The second attribute is the $path$ where the transformation method is placed in. Finally, $parameters$ are the arguments that the transformation method receives for execution.

   (c) **Load (L):** Indicates where the S_i processing results are delivered. In this phase, two attributes are considered: $< type, path >$. Where $type$ is the output type that provides S_i (folder, file), and $path$ where the processing results are loaded, and which are then the input to the service S_{i+1}.

6. **Patterns (Pa):** The pattern considered in this research work is the **Manager/Worker (Pa_Ma/Wo)** pattern. This pattern has a Manager in charge of distributing the workload (data to processed) among N Workers. The Manager divides the original data set and assigns it to each of the available workers to process the data source (DS) in parallel.

The elements of the interconnection model that describe the verifiability network are:

1. **Organization (O):** For the workflow component (W), an organization (O) represents an entity that deploys a set of services ($S_1, S_2, ...S_N$). In the proposed model, these services are described as a stage ($St = \{S_1, S_2, ...S_N\}$) that is part of the processing of the data in a workflow (W) in which participates multiple organizations. In this sense, an organization is an entity in charge of one or more stages that belong to a workflow ($St \in W$).
2. **Nodes (N):** From the verifiability network point of view, an organization is in charge of a set of Nodes (N) that will be used to create a blockchain network. Each node contains an API REST through which an organization records the transactions it performs in its stages (St) that belong to a workflow ($St \in W_i$). The Nodes can acquire any of the following roles:
   (a) **Peers nodes (N_{peers})**: Those are the nodes of the verifiability network that issue transactions.
   (b) **Peers by organization (PxO):** are the peer nodes (N_{peers}) will the system deploy on the IT infrastructure of the organization (O). For simplicity purposes, this parameter has the same value for all organizations along the use cases. Thus, the Peers nodes

(N_{peers}) can be computed easily. For example, if you have a configuration of 2 organizations $Org = \{Org_1, Org_2\}$ and $|N_{peers}| = 2$, it means that each organization will have two peer nodes, resulting in 4 peer nodes in the verifiability network. The general formula is given below:

$$N^\circ\ Peers\ nodes = \underbrace{|Org|}_{N^\circ\ Organizations} * \underbrace{|N_{peers}|}_{N^\circ\ Peers\ by\ Org} \tag{3.1}$$

(c) **Orderer manager node (N_{or})**: establishes a consensus mechanism between the peer nodes when verifying transactions. The order manager node also verifies the sequence of transactions and records consistency.

(d) **Certification Authority Node (N_{ca})**: manages the certificates and private keys delivered to all the peer nodes belonging to a specific organization.

3. **Transaction (Tx):** It represents an action performed by a service (S) in a workflow (W). The structure of a transaction (Tx) depends on the business network (BN) that was defined.

4. **Blockchain (BC):** is a *peer-to-peer* network or P2P, where are recorded the transactions performed by each service (S).

5. **Business Network (BN):** defines the business logic ruling the transactions performed by the participants in a workflow, as well as the abstraction of the entities and assets included in those transactions.

6. **Administrator (Ad):** manages the access keys associated required for a given organization (O) to make register in blockchain (BC).

Once the declarative model of interconnection for the IoT dataflow and the verifiability network are described, the Global Manager, through a Construction Manager and an Operation Manager, can build and operate a CD/CV system.

3.2.2 Construction Manager

This Manager has three main components: Orchestrators, Launchers, and Choreographers. They work together to build a CD/CV system from the ETL interconnection model and *DAG* scheme described in the previous section.

Construction of the CD/CV system Two main phases are part of the construction of the CD/CV system: a preparation phase, and a deployment phase.

Preparation phase: Interconnection ETL model In the preparation phase, the Construction Manager creates the ETL configuration by using the information delivered by end-users about services at the stages of a workflow ($St = \{St_1, St_2, ..., St_N\} \in W$, where $N \rightarrow$ Number of stages). The user (administrator) defines the number of services that will be executed in each St_i as well as the sequence of each $St_i \in W$. The organizations also can define the parallelism degree (services per stage) to improve the stage and workflow performance, which is expected to be used to compensate for the overload produced by the registration of transactions performed by each stage. The number of services and applications are declared in the ETL interconnection model as:

$$St_i = Ns_i \rightarrow \text{Number of services in stage i}, \; i = 1 \ldots N.$$

The CD/CV Construction Manager creates an architectural pattern using ETL information (see Figure 3.4) to materialize the workflow (W).

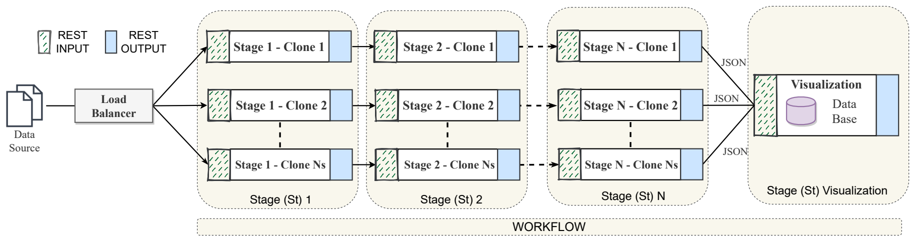


FIGURE 3.4: The architectural pattern of the CD/CV Manager: an example of a service for the analysis/processing of data from IoT environments.

The applications and services of each stage are encapsulated into containers for organizations to clone and deploy stages on IT infrastructure.

For simplicity, we can say that the CD/CV schemes include two parts: the CD focused on stages of the workflow, and the CV focused on nodes of the blockchain network. For this thesis, in the CV part of the schemes, the number of organizations participating in a workflow corresponds to the number of stages of that workflow. Nevertheless, the CD schemes can create multiple associations (e.g., multiple stages compose one organization or multiple organizations compose one stage).

The CV part includes the following components: *1)* By default, a single order manager node is defined to establish consistency in transactions although multiple order manager nodes can be defined to conform the ordering service. *2)* A CA node for each organization to manage the cryptographic material for each stage to register transaction; and *3)* The number of peer nodes N_{peers} defined for *P2P* of the blockchain. This parameter is assumed to be a variable for experimental purposes and of course in production depending on the IT-resources.

In this sense, the end-user delivers information about the CV scheme, such as the number of organizations and the number of peer nodes per organization in the verifiability network. These parameters are defined in the ETL interconnection model as follows:

$$\textbf{PxO} = N_{peers}$$
$$\textbf{Orgs} = \{Org_1, Org_2, ..., Org_N\}$$

Where N_{peers} is the number of peer nodes for each organization, and $Orgs$ defines participating organizations. This research assumed that there is a correspondence between the number of workflow stages and the number of participating organizations.

The total number of nodes in the verifiability network is represented by the following notation:

$$\textbf{Nodes (N)} = \{Peers_{Org1}, Peers_{Org2}, ..., Peers_{OrgN}, Nodes_{CAs}, Nodes_{ord}\}$$

Where:

$$Peers_{Org_i} = \{peer0_{Org_i}, peer1_{Org_i}, \ldots, peerN_{peers_{Org_i}}\} \in Orgi \text{ con}$$
$$i \in 1, 2, \ldots, N$$
$$Nodes_{CAs} = \{ca_{Org_1}, ca_{Org_2}, \ldots, ca_{Org_N}\}$$
$$Nodes_{ord} = \{N_{ord1}, \ldots, N_{ordP}\} \rightarrow \text{Nodes to establish consensus.}$$

By default, an $administrator$ for each organization is defined for each CD/CV scheme. Administrators are responsible for adding new participants to the CD/CV schemes and to associate stages to organizations.

The general formula for estimating the number of nodes deployed is shown below:

$$N^{\circ}\ Deployed\ C_o = \underbrace{|N_{ca}|}_{A} + \overbrace{2 * (N^{\circ}\ Peers\ nodes)}^{B} + \underbrace{(3 * N_{or})}_{C} + \overbrace{(N_{cli})}^{D} + \underbrace{1}_{E} \tag{3.2}$$

Where **A** is the number of Certifying authority nodes; **B** is the number of Peer and DB nodes (for each peer node there is a database node); **C** is the number of Consensus nodes (for each orderer node two auxiliary nodes are deployed: Kafka and zookeeper); **D** is the number of Client nodes; and **E** is one container where the intelligent contract is initiated.

Configuration for the CD/CV scheme deployment. The Construction Manager creates configuration files for the deployment of both components: continuous delivery (CD scheme part of the workflows) and the continuous verifiability network (CV scheme part of the blockchain) on a given infrastructure. These files create the paths of Extraction (from data sources), Transformation (execution of applications and services), and Loading (sink) to a shared volume in the virtual container of the stages (CD) and nodes (CV). The definition of paths in each component of the CD/CV schemes and enables the Construction Manager to materialize the CD/CV schemes to realize workflows (managed by CD scheme part) and the corresponding blockchain network (managed by CV scheme part).

The following paths are added to the continuous verifiability (CV scheme part): *1)* Path to Hyperledger Fabric, for the nodes to get the binaries required to create the cryptographic material (certificate, and public/private keys) as well as the binaries required to generate the building artifacts of the blockchain (peer communication channel, genesis block, and membership manager); and *2)* Path to business module, which points to the configuration file containing the business model associated with the ETL model of all the stages of the workflow.

Using CD/CV schemes during executing time This section describes three subcomponents such as *Orchestrators*, *Launchers*, and *Choreographers*, which are used by CD/CV Construction Manager to start the execution of the deployed stages of workflows (CD virtual containers) and nodes of blockchain network (CV virtual containers). These components ensure the establishment of controls over the registration of the transactions performed in stages of workflows in the blockchain nodes.

The *Orchestrators* enforce the ETL interconnection by creating configuration files that are used by *Launchers* to initiate all the deployed stages of workflows (CD scheme part) and blockchain nodes (CV scheme part). Finally, the *Choreographers* start the injection of workload (data for stages and transactions for nodes), and the execution of the applications of the stages and the connectors of these stages with the blockchain nodes.

These connectors are services/applications that are configured (environment variables, cryptographic material, etc.) to issue a transaction on behalf of an organization on the blockchain from the data obtained at each stage of the workflow. Each connector represents a client/interface between the CD and CV component of a given organization.

In detail, the Orchestrators create the configuration files and artifacts needed to deploy the CD/CV schemes for workflows. Two Orchestrators were developed: the CD Orchestrator, in charge of continuous delivery, and the CV Orchestrator, designated for continuous verification of transactions.

The CD Orchestrator creates the necessary settings to synchronize the applications to communicate with each other. To do this, it receives the representation of the network from the Manager (i.e., a *DAG*) and sends orders to CD Launcher for coupling the stages following the *DAG* by configuring input and output ports in a given infrastructure for each stage.

The CV Orchestrator creates the logical artifacts needed to create the verifiability network for the workflow defined in the CD scheme. These artifacts implement the business model of each pair of stages in the verifiability network (e.g., the definition of participants and assets) for Launchers to create the smart contracts and to prepare the cryptographic components (keys for connectors in stages to make transaction registration) as well as to couple the *P2P* network with the CA (certification authority).

The CD Choreographs starts the injection over the stages following the *DAG* over the deployed virtual containers (stages) to guarantee the correct and valid coupling and interconnection of the applications deployed on the virtual containers of the workflow stages.

The CV Choreographs verify that all the connectors of the stages can reach the blockchain nodes, that the cryptographic components (keys for accessing to the blockchain network) are valid, that the smart contracts have been successfully created, and the transactions of the applications executed at the stages (workflows/pipelines) can be registered under the control of the contracts.

3.2.3 Operation Manager

This module is responsible for managing the proper functioning of the IoT dataflow and the verifiability network. The Operation Manager must ensure the continuous delivery of the data in the workflow and also the continuous verification of the transactions performed. Figure 3.5 illustrates the Management Framework once it is in operation.

Figure 3.5 shows an example of a workflow including *N* stages containing *n* Virtual CD containers that will process a data source and extract, from it, knowledge, or information useful for decision-making in organizations or

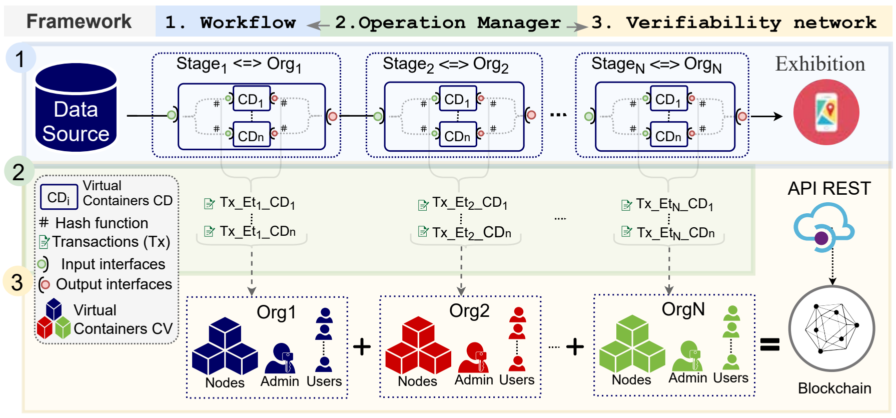


FIGURE 3.5: Management framework CD/CV in operation.

interested entities. At this point, the continuous verifiability component starts recording the transactions (Tx) performed by the stages of the IoT dataflow in the verifiability network. Summary functions (or commonly known as *hash* functions). The transactions are registered by using signature (hashes), which produces an identification for a content. A transaction includes the hashes of contents incoming to the stages and the resultant contents transformed by the applications executed at a stage, as well as the paths used to extract, transform, and loading contents.

This mechanism ensures that any alteration in the input contents and the resultant contents. This means, the contracts verify the hashes of the assets before and after to be transformed by each stage in a IoT dataflow; as a result, the end-users can verify the integrity of the data at each stage of the workflow. Moreover, the chaining of the ETL paths and hashes allows end-users to detect/prevent from stages processing unexpected contents by querying to the blockchain network.

3.3 Integration of GM-CD/CV with scientific workflows

The integration of GM-CD/CV to existent scientific workflows engines for processing data from IoT is feasible.

1. Workflow Stages: GM-CD/CV requires that each workflow stage is in the form of an executable service or application. This restriction is because the initial design of the Global Manager CD/CV delivers as a response from a i stage to a $i+1$ stage, the output of the server where the service is exposed (API REST response), or the logs of an executable application.

2. Declarative process: GM-CD/CV assumes that the user declares the ETL of each application.

3. Data source availability: GM-CD/CV requires the location of a data source (set of tasks to be processed) to start the continuous delivery process.

4. Data Exchange: GM-CD/CV requires that the results exchanged between stages are packaged in a standard format such as JSON to ensure continuous delivery. It was assumed that each organization that performs the exchange known the data placed in the exchange file.

5. Verifiability Network: GM-CD/CV already incorporates the blockchain manager. If the user needs to change this blockchain manager, he will have to adjust the GM-CD/CV deployment engine (i.e., use Ethereum or another mechanism to use cryptocurrency in transactions).

3.4 Periodic Contract Verification

Most operations executed in the blockchain consists on logging the status or values of certain data. However, smart contracts can be associated to events to validate rules, verify values, or firing new actions. Before proceeding with contract verification procedure, the two elements considered to carry out the verification process are defined: 1) the penalty clauses; and 2) the type of verification to be performed.

1. A **Penalty Clause** is defined to adjust the value of the contract established according to the values received from each sensor and the compliance with the restrictions established in the contract. A penalty clause is declared per sensor in the model and it is defined as the following three-element function $Pen_i = \{x, cond, penalty\}$, where x is the value received in real-time from the sensor, *cond* is the conditional function that must be met to record the value of x as valid, and *penalty* determines the value of the penalty to be deduced from the total value of the contract.
2. **Type of verification**. The verification process consists of contrasting the data acquired and registered in the blockchain, with the restrictions stated in the contract for such data. It also applies the penalty clauses if required. The proposed system incorporates three types of contract verification:
   (a) **Verification by Identifier**. In this case only one record identified by a unique id in the network is analyzed. This type of verification is useful when the origin and time of the failure is known and it is conceived to verify the restrictions and possible penalties for a single record in particular.

(b) **Lot Verification**. Since all records in the network have by default the time stamp for transaction creation, the lot verification is meant to recover all records in the blockchain which time stamp is within a specified time interval. This interval could be previously defined by the user or could be calculated automatically to execute the verification periodically.

(c) **Results verification**. This type of verification is thought to reuse the results of previous verifications and can be useful in subsequent queries for different purposes and reduce costs of running again previously executed verification processes. As different actors may need to verify the blockchain records, and since the records in the blockchain are immutable, it does not make sense to repeatedly run verification processes on already verified set of records as they would always give the same results.

3.4.1 Contract verification process

The contract verification process consists of recovering the data stored in the blockchain network, that have been registered by the different participants or entities of business logic, and subsequently, determining if each of the records accomplish the contract terms, which have been established in the definition phase by the user administrator. In case of failure to comply with the constraints, the system proceeds to execute the penalty clauses established on the final value of the shipment (see Section 3.2.3). Figure 3.6 shows the verification processes that are executed to determine whether a contract is valid or has violated the restrictions.

In most blockchain applications for transport reviewed, contract verification is made once the products are delivered o whether there is a contract breach in the delivery date. This may be due to business logic, but in most cases it is

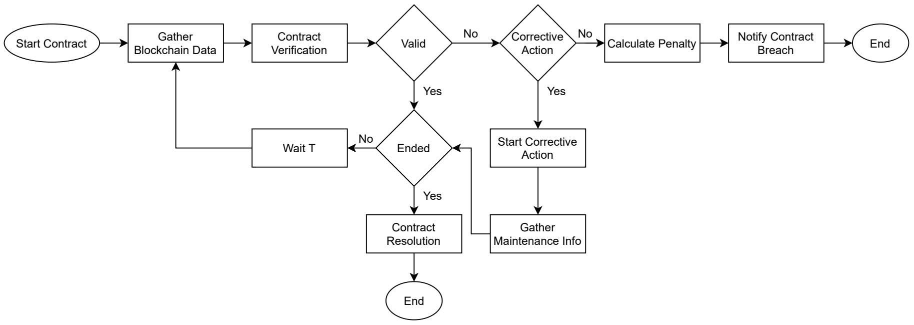


FIGURE 3.6: Continuous contract verification flowchart

due to the fact that data are not reported in real-time to the actors and then they cannot verify the contract. In our case, our solution is able to collect data in real-time and submit the transactions to the blockchain network so that data are registered immediately or with a small delay. Having data registered in real-time, we propose to run a continuous contract verification process, with a periodicity that can be defined by the business logic.

Initially, the smart contract (C) is created and activated only after the entire blockchain network has been established and a communication channel has been established. Once this happens, the contract can begin collecting data or initiating transactions. For instance, let's consider a food logistics scenario where the contract is linked to each shipment made by a truck. The contract is activated as soon as the truck embarks on a new route, and it is required to validate the contract for each data received by the truck's sensors. After that, the sensors (Se) on the truck start collecting real-time data, which is then recorded on the blockchain network.

Simultaneously, when the contract is started, a periodic verification process is also started for the contract. This verification process retrieves data from the network for analysis and every time (T), being T a period defined for every shipment, determines if the restrictions $Res \in C$ are fulfilled. If the restrictions are satisfied, a successful verification process is recorded for this period. Since the duration of a shipment will be, in general, longer than the verification period, the process will be repeated for the series of subsequent periods in the shipment. This procedure is executed repeatedly in a closed cycle until the status of the whole shipment is delivered. In other words, the truck has reached its destination and the product has been delivered to the final consumer. If all the records of the verification process are correct, then no alert is issued, and the resulting record of the verification will be a valid contract. In this case, the contract is deemed to have been satisfactorily terminated.

To achieve the closed cycle between the data capture and the periodic verification procedure, the administrator user in the definition phase establishes the periodicity parameter T, which is the period in which the contract verification process is executed. Any record in the blockchain written during this time interval and related with the contract will be verified. The result of each periodic contract verification is also recorded in the blockchain, as a new record that includes the results of the verification performed. This approach has two main advantages: near real-time status of each shipping and scalability in contract verification.

As contracts are verified every T for a shipping, a periodic verification process might determine in near real-time that the contract verification is not valid in the interval analyzed $[t-i, t_i+T]$ due to errors in data capture or conditions breaching the contract. In this case, the error condition is also recorded in the blockchain to ensure non-repudiation and the participants are notified of the incidents found to immediately take corrective measures (if possible). After that, the system evaluates if the contract has ended. In case not, it continues the periodic contract verification until the delivery of the shipment. Otherwise, if it is not feasible to apply the corrective or maintenance measures, the penalty clauses established in the contract are executed. Following this procedure, and depending on the violations committed, the final value of the contract is established, and the breaches of the contract are notified to the interested parties. The penalty clauses are part of the proposed system model and were described in Section 3.2.3. The process of verification and application of sanctions is summarized in Algorithm 1:

Algorithm 1 : Continuous contract verification

```
 1: T = verification period (seconds)
 2: cts = current timestamp (“dd/mm/yy-hh:mm:ss”)
 3: pts = timestamp with T seconds before cts (“dd/mm/yy-hh:mm:ss”).
 4: for All $Tx_{se}$ recorded between [pts, cts] do
 5:     $C_i$ = IdentifyContract ($Tx_{se_i}$)
 6:     $Se_i$ = IdentifySensor($C_i$)
 7:     $Res_i$ = GET $C_i$ restrictions for $Se_i$
 8:     if $Tx_{se_i}$ does not comply with the $Res_i$ then
 9:         $Pen_i$ = GET Penalty clauses of $C_i$
10:         value $C_i$ = CalculateContractValue($Pen_i$,$Res_i$)
11:         Update result
12:     end if
13:     Update result
14: end for
```

Every result of a verification process is saved as a new record in the network, in this sense, a result variable is declared (step 1) that will store all the transactions that do not comply with the established restrictions. By default, the proposed system executes the verification process automatically every verification period (T) in seconds (*step 1 pseudocode*). Then, the current system time *cts* is retrieved (*step 2*) and the time stamp of the next verification interval is calculated as pts = timestamp of T seconds before that cts (*step 3*). The variables *pts* and *cts* denote the initial time and end time respectively of the time interval to be analysed. Then all blockchain records that were created for the shipment of the contract within the previously calculated time interval are analysed one by one (*step 4*). For each record, its contract is identified (*step 5*) and the sensor identifier is extracted (*step 6*) which collected the data sent in the transaction. Knowing the sensors involved in the execution of the current C_i contract, the restrictions defined by $Res \in C$ that must be met are consulted (*step 7*). If the current transaction (Tx_{se_i}) does not comply with the restrictions of the contract (*step 8*), the penalty clauses are consulted to apply them to the final value of the contract price (*steps 9 and 10*) and the *result* variable records the current transaction and an invalid contract status as a warning for subsequent verification.

The proposed procedure increase scalability as the verification is made for a small subset of transactions (depending on T) and the result is stored. Thus, on the one hand if all steps are correct, the final verification process is automatically solved. On the other hand, if there was a contract breach in a cycle, the verification processes do not need to analyze again all the data, are the verification results of every steps are also logged in the ledgers, providing non-repudiation. Even if a third actor would like to verify the whole shipping, she only had to verify the results of the periodic verification processes.

3.5 Summary

In this chapter, a new approach for the development of CD/CV (**C**ontinuous **D**elivery/**C**ontinuous **V**erifiability) systems that incorporate blockchain technology with workflow engines and/or pipeline builders is presented.

The GM-CD/CV (or **G**lobal **M**anager - **C**ontinuous **D**elivery /**C**ontinuous **V**erifiability) presented in this research work is an intermediary manager between workflow engines and blockchains for verification and validation of the execution of IoT dataflow in an established sequence and the integrity of digital assets. The CD/CV schemes proposed allow performing continuous verifiability based on blockchain (CV model) of the transactions performed by applications included in workflows (CD model), that are recorded transparently and automatically in the blockchain system. These schemes incorporate parallel patterns encapsulated into virtual containers to reduce the impact of such verifiability on the performance of workflows when producing IoT dataflows.

I have explained how it is possible to create a CD/CV system by going through the stages of preparation, deployment, and operation using a particular technology. The Global Manager (GM) prototype, which includes two major components - the Construction Manager and Operation Manager - was presented as an example. Additionally, a model was used to define the delivery (CD) and verification (CV) components, which will be used to create the CD/CV system. Based on the features of this model, guidelines have been established for integrating the GM-CD/CV technology with scientific workflows.

Lastly, suggestions and detailed instructions are provided for a periodic contract verification procedure that should be implemented once the CD/CV system is active and data gathering is underway.

CHAPTER 4

BLOCKCHAIN OPTIMIZATIONS

4.1 Introduction

The methodology proposed in the previous chapter deploys two workflow chains mixing Internet of Things (IoT) and blockchain. However, using blockchain technologies for IoT registration has currently considerable challenges, such as transactions performance, scalability, and near real-time contract verification. Trying to cope with some of them, in this chapter, we propose three platform independent optimization techniques (atomic transactions, grouped validation and high-performance delivery) that enhances data transactions protocol and the data storage procedure in blockchain, which allows to take corrective actions to reduce operational costs and increase benefits in current applications.

4.2 Real-time data acquisition

IoT are systems generate massive amounts of data, specially if they collect high-frequency real-time information from different types of sensors. One of the main challenges in the usage of blockchain technology for IoT workflows is the time required to process and confirm a transaction on the network [117], since all participants must approve the transactions. Therefore, if you do not have an efficient system to process transactions, the scalability of the system could be compromised. The challenge is much greater if, in addition to considering a scenario where you have a secure supply chain (incorporating blockchain technology), you consider a high-demand supply chain (incorporating IoT devices) to obtain all the benefits this entails. Achieving the adoption of these new technologies to traditional supply chains is not trivial. Firstly, the IoT environment devices can capture large volumes of information in short periods, This generates a challenge in the infrastructure required to analyze and manage the captured data, which is not prepared in many companies [50]. Secondly, the volume of information generated by the IoT devices incorporated in the supply chains directly affects the number of transactions that need to be registered and verified in the blockchain, generating scalability and performance issues [118].

Some recent works have contributed to the development of optimization techniques to increase performance of blockchain technology [92, 94, 119]. However, these optimizations are platform-specific and do not contribute to a generic solution. Other research works have proposed platforms and systems that allow verification of supply chain processes and data [57, 73, 75, 81]. However, these works consider the verification process in the final stage of the supply chain, which prevents, limits or delays the use of corrective measures to cope with failures in the supply chains. A last concern is that many companies lack the necessary architecture (objects, networks, data services).

This section presents the design decisions and optimizations proposed in our work to manage and store the data produced by the IoT devices in the blockchain network. Management and data storage is implemented after the blockchain network and the contract logic have been fully deployed, as shown in Figure 4.1.

The use of blockchain technology for information management has been proposed because it allows the decentralization of information, preventing a single point of failure, ensuring the persistence, anonymity, and auditability

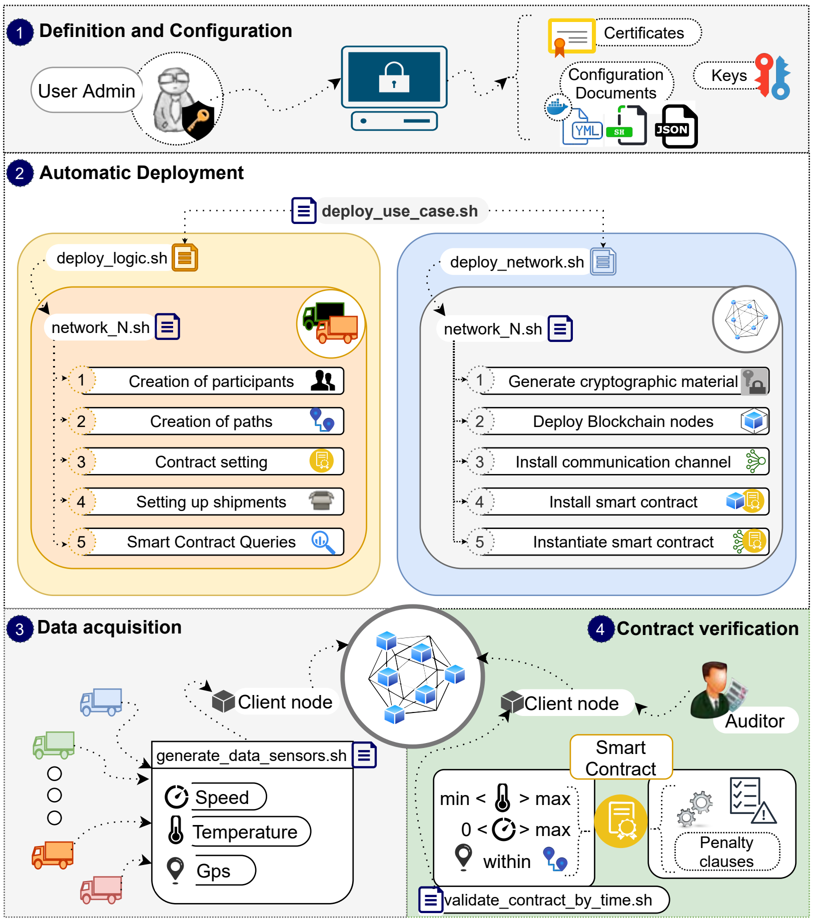


FIGURE 4.1: Definition phases of the designed model: definition, deployment, acquisition and verification of data

of the data. Those are critical points of failure in the security and reliability of IoT devices, because they have limited hardware and therefore cannot implement security mechanisms that require a lot of resources, making the data unreliable. However, as IoT devices are capable of sending high volumes of data in a short time, the design of the network should be scalable and have a high throughput to be able to manage the data in real-time, preventing data loses or delays in storage.

4.3 Traditional blockchain implementation

As a first approach to the problem, the usual way of updating a block in a blockchain was used, which means concatenating records in a block per sensor. In other words, when a sensor takes a new measurement, all the measurements previously taken by that sensor must be recovered, new data are added to the existing measurements in a new block and, finally, the transaction is stored in the ledger. This methodology allows to simplify data recovery when a query is executed, because it is only necessary to get the last transaction, also called *world state*, realized in a specific event by the sensor whose measurements are consulted (e.g. temperature). This query can be carried out because the last transaction contains all the values read by the sensor, until that moment.

A case study described in Section 5.3 has been defined to evaluate the traditional strategies and proposed techniques. Using this case study, two experiments were conducted with this concatenated records solution to measure performance (see Section 5.3.7) and verify the correctness of the performance (see Section 5.3.7) when building a blockchain network for the defined use case.

The case study described in Section 5.3, focuses on these two aspects as they are critical metrics of the blockchain network, considering the need to cope with a massive amount of real-time data.

To explain the optimization techniques being proposed in the Section 4.4, the subsequent section outlines the essential steps and elements involved in the implementation of these optimization techniques.

4.3.1 Hyperledger Fabric implementations

Although the implementation of blockchain technology in Hyperledger fabric has extensive documentation[1] covering various aspects such as components, models, architecture, etc., this section specifically focuses on the components and processes relevant to the proposed techniques.

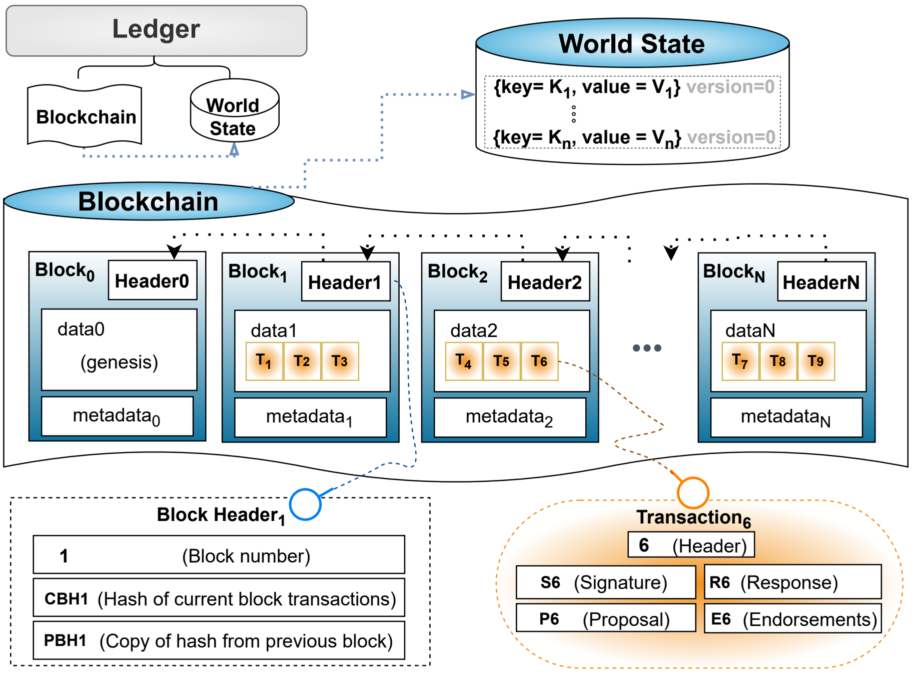


FIGURE 4.2: Ledger structure

The components of Figure 4.2 and the essential elements in the proposed techniques are described below:

1. **World State (WS):** It is a database that holds the latest values of a specific set of states or versions of the *ledger*.

[1]https://hyperledger-fabric.readthedocs.io/en/latest/whatis.html

2. **Blockchain (BC):** It is a log of transactions that documents all modifications that lead to the current *world state*. BC determines WS. *Transactions* are grouped together into *blocks* that are added to the *blockchain*. Unlike the *world state*, the blockchain data structure is **immutable**, meaning that it cannot be altered once written.

3. **Ledger (Le):** A ledger consists of a world state (WD) and a blockchain (BC). The network preserves several duplicates of a *ledger* and ensures that each copy remains consistent with the others using a mechanism known as **consensus**.

4. **Smart Contract:** is a file that includes the functionalities required (ledger APIs to get, put and delete states) to carry out the business logic (queries and functions definition). A smart contract creates the proposed ledger update.

5. **Chaincode**: In Fabric, smart contracts are implemented using chaincode, a type of logic that interacts with the ledger. This chaincode is created using the chaincode APIs.

6. **Transaction:** a transaction captures changes to the *world state*. Every transaction comprises five essential components, including:

    - Header: This section includes transaction metadata, such as the name of the associated *chaincode* and its version.
    - Signature: Refers to a cryptographic signature generated by the client application, which serves as proof that the transaction data has not been altered. This section necessitates the application's private key to produce the signature.
    - Proposal: This component encodes the input parameters that an application feeds into the *smart contract*. During the smart

contract's execution, this proposal serves as a set of input parameters that, along with the current *world state*, determines the resulting world state.

- Response: This section includes the values of the Read-Write set (RW-set) that represents the world state before and after the transaction execution. It acts as the output of a smart contract, and if the validation process is successful, it will be applied to the general ledger to update the world state.
- Endorsements: This component is a catalog of signed transaction responses from each organization that fulfills the endorsement policy's requirements. Although only one transaction response is present in the transaction, several endorsements are included. Each endorsement encodes the transaction response specific to your organization. If the endorsement policy is not satisfied, the transaction will be deemed invalid and will not modify the world's status.

7. **Peer:** The peers stores and manages copies of ledgers and smart contracts.

8. **Channel:** A method through which elements within a Fabric blockchain network (peer nodes, orderer nodes, applications, etc) exchange information and conduct transactions in a confidential manner.

9. **Policy:** Fabric policies define the rules and procedures that govern how members reach consensus on whether to approve or reject modifications to a network, *channel*, or *smart contract*. Each smart contract contained within a chaincode package has an endorsement policy that outlines the number of *peers* from various *channel* members required to validate and execute a transaction for a particular smart

contract. Therefore, endorsement policies determine which *organizations*, represented by their peers, are responsible for endorsing or approving a proposal's execution.

10. **Transaction validation process:** In Hyperledger Fabric, the transaction validation process consists of the following steps:

    (a) Proposal: A client sends a transaction proposal to the nodes in the network.

    (b) Endorsement: The endorsing nodes execute the relevant smart contract code and, if the proposal is valid, generate an endorsement. This endorsement is a digital signature indicating that the result of the transaction execution is correct.

    (c) Endorsement collection: Endorsements from endorsing nodes are sent to the client.

    (d) Verification of endorsements: The client verifies that it has received enough valid endorsements from endorsing nodes as specified in the endorsement validation policy.

    (e) Sending to the orderer nodes: The client sends the transaction and the corresponding endorsements to the ordering nodes. The orderer nodes receive the transaction and validate it again. If the transaction is valid, they include it in a block along with other transactions. The ordering nodes then use a consensus algorithm to agree on the order and content of the blocks.

    (f) Confirmation: Once consensus has been reached, the block is confirmed by the network nodes and added to the distributed ledger, which finalizes the transaction.

It should be noted that, depending on the network configuration, there may be several endorsing and ordering nodes involved in the validation process of a transaction in Hyperledger Fabric.

11. **MaxMessageCount:** In Hyperledger Fabric the transactions can be validated individually or in blocks of transactions, by modifying the parameter *MaxMessageCount*. It means that, instead of validating immediately each transaction, the system waits until a certain number of transactions is available (e.g. 10) to make delayed validation. This feature allows the system to make optimizations, however the problem is that until a transaction is not validated, it will not be stored in the ledger. Thus, it is not available to other partners in the blockchain network, creating delays in information availability.

A set of experiments (see Section 5.3.7) was conducted to analyze the performance of the blockchain system under this traditional implementation and from which possible implementations of improvements described below are observed or determined.

4.4 Proposed Optimizations in Blockchain

As shown in the previous section and reflected in the evaluation carried out in Section 5.3.7, there are problems to be solved in order to use the blockchain technology in a network gathering real-time data from IoT devices. Those problems are related to performance, scalability, and possible data losses when transactions are processed. The former problems have been detected in previous works, but most solutions proposed to cope with them try to enhance the Internals of the blockchain platform used (validation, protocols, etc.).

For the sake of usability, and to increase adoption of the solution, two generic optimizations in transaction processing are proposed in this thesis that are compatible with any blockchain platform used to implement the blockchain network. The optimizations do not require modifying the internals platform for them to take effect. These improvements are: using atomic records to store the data sent by the sensors and changing the validation block size.

4.4.1 Atomic Records

This first optimization proposed in this research work consists of storing each transaction carried out by a sensors in an atomic form, creating a new block for each transaction from a sensor. This block includes the minimum information needed for the transaction to be executed. For example, their content can be the sensor identifier, the value of the measurement taken, and the identity of the truck where the sensor is installed. For this purpose, when a sensor sends a new transaction, a new transaction is added for this sensor after the most recent one. This avoids the need of recovering all values previously stored by the sensor to concatenate the new value afterward, as it happened in the model described in the previous lines.

The effect of this optimization is critical, as the size of the transactions made by each sensor remains constant over time and the transaction replication time in all the peer nodes is also constant, as it only contains the last value and the minimum necessary information. Thus, scalability should be ensured. As the blockchain technology provides immutability and auditability, we provide the functionalities needed to get the history of values of a given sensor since it was created. Therefore, it is possible to execute queries to see the content of the last transaction, a set of transactions, or to recover all the transactions executed, which is also an advantage over the previous model that always returned the complete history of a sensor requiring further post-processing.

The performance evaluation when applying this optimization is described in the case study (see Section 5.3.7).

4.4.2 Changing the Transaction Block Size

To cope with the delays in the data availability, we proposed a second optimization technique in our model: modifying the maximum number of transactions that can be stored within a validation block. It also uses atomic transactions to ensure that they are independent of each other. This optimization allows increasing the performance of the blockchain network since the validation process (the phase that produces the highest latency) is not carried out every time a transaction is made, but by groups of transactions, which reduces the number of times the validation process is performed on the network.

The performance evaluation when applying this optimization is described in the case study (see Section 5.3.7).

4.4.3 Artifact generation and automatic deployment

This section outlines a new approach to generate and deploy blockchain network artifacts, considering the multiple components (such as nodes, certificates, private keys, identities, channels, policies, etc.) needed to create a verifiable network using Hyperledger Fabric and deploy it within a specific infrastructure.

In order to set up a functional verifiable network that can record transactions, certain steps need to be taken. These steps include creating cryptographic material for each component of the network, defining the environment of each node, establishing communication channels, creating smart contracts, and installing them on the relevant nodes. These steps require technical expertise on the part of the verifiable network administrator. This could be a problem if the participants or organization wants to incorporate a verification component into their processes but lack the technical knowledge needed to deploy a verifiable network. Furthermore, if there are numerous organizations with many participants involved in an IoT scenario, the configuration of the verifiable network may become impractical or not easily scalable.

The proposed solution aims to address the limitations mentioned above by automatically generating the essential components required for verifying IoT data flows. This is achieved through a series of processes or abstractions, which include:

1. The number of organizations in the workflow is determined by the number of stages, and the user can define the number of peers for each organization.

2. A communication channel is established to facilitate the workflow.

3. An orderer node is created to manage the transaction order for the entire network.

4. The smart contract logic follows the ETL model and is pre-defined. Assets refer to the files or data processed through the workflow, while participants refer to the programs, services or processes that receive and generate new data. The hashes of incoming and outgoing files are part of the smart contract logic.

5. The configuration of environment variables, ports, and routes for each verifiable network node is based on the available infrastructure where the network will be deployed, and not limited to localhost deployment.

6. The proposed method generates all the necessary cryptographic material by taking in the user's ETL declaration and the desired number of peers for each organization. This is achieved by defining the value of the Npeers variable in the ETL model. For example, if the user's ETL declaration specifies a 3-stage workflow ($w = \{stg1, stg2, stg3\}$) and 2 peer nodes are needed per organization ($N_{peers} = 2$), the proposed solution automatically creates the cryptographic material for 3 organizations, with each organization having 2 peer nodes.

The abstractions mentioned earlier enable the automated and generic construction of a blockchain network based on user-defined parameters, without the need for technical expertise.

The research work utilizes the automatic generation and deployment mechanism in each of the case studies evaluated. Given that the number of components (e.g. organizations and peers) of the blockchain network differs in each case study, it is crucial to redefine the components and artifacts of the network in every scenario.

4.4.4 High-Performance delivery

The methods explained earlier can be helpful when real-time IoT data is directly recorded in a decentralized database of a blockchain network, and then analyzed in the context of a smart contract that is designed for a specific use case. For instance, in an environmental setting, the smart contract can assess temperature data to determine if it falls within acceptable temperature limits and predict potential fire risks. Similarly, in a transportation scenario, the smart contract can evaluate speed data to verify if it meets the permitted speed limits and detect any traffic violations.

However, there are situations where a single piece of data is not enough to trigger the smart contract or business logic to take action. For instance, in a healthcare setting, a device that captures the electrical activity of the heart (or electrocardiogram) may provide data that requires more than one piece of information to determine a possible irregularity in the heart rhythm. Typically, algorithms designed to detect anomalies require input signals at specific time intervals. In this case, the device's sampling frequency can generate thousands of data points in a few minutes, as it can range between 500Hz and 2000Hz. In fact, the higher the sampling frequency, the more precise and high-quality the recorded signal becomes.

In this situation, several data sets or structures are produced that must be maintained in their original format for future processing, analysis, or storage in the blockchain.

Within the proposed solution outlined in Chapter 3, the CD component produces results in a JSON format at each stage of the workflow (see 3.3). However, managing a large amount of data generated by a single stage of the workflow in this format can be challenging, particularly in scientific workflows. Moreover, this format does not permit or restrict access to data by multiple parallel processes.

HDF5 (Hierarchical Data Format version 5) is a technology that is commonly used to store and share scientific and engineering data [120]. The development of this technology was driven by the need for a file format that could handle large scientific datasets and store them efficiently while allowing for easy access. The key features of HDF5 include platform independence, optimized storage techniques, support for managing large amounts of data, hierarchical organization of data, metadata support, and a simple API for data access.

Based on the characteristics of HDF5 technology discussed earlier, we think that integrating HDF5 as a way to save the outcomes of each step of the CD component in the proposed solution can greatly decrease the amount of transactions produced on the blockchain. This would be achieved while preserving all essential information within the business logic, without any loss of pertinent data.

The primary CD/CV prototype plan involves adopting a fine-grained approach, where each file produced from the workflow stages will generate one blockchain transaction. In contrast, the coarse-grained approach will involve creating one transaction for the entire group of files created by each stage within a specific time period.

By incorporating HDF5 technology in the CD/CV prototype, the number of resulting files in the workflow can be decreased and the size of the transactions that must be generated in the verifiability network can be reduced. Furthermore, the parallel I/O functionality allows us to save the outcomes of numerous threads from the same workflow stage simultaneously in a single file. Consequently, the information from multiple processes (or workflow stages) can be stored in a single transaction on the blockchain network using the resulting file information.

4.4.5 Enhancing the continuous delivery component.

To leverage the parallel I/O capability of HDF5 and achieve improved data transfer performance between workflow stages (component CD), an alternative solution to the one illustrated in Figure 3.4 from Chapter 3 is proposed in this section. This alternative solution involves integrating DIY [121], and Lowfive technologies into the software stack of the proposed prototype CD/CV.

1. **DIY**: DIY is a library[2] that allows for the implementation of scalable algorithms through block-parallel processing, which can be executed both in-core and out-of-core. It offers the flexibility of using a single program with one or more threads per MPI process, thereby combining distributed-memory message passing with shared-memory thread parallelism in a seamless manner.

2. **Lowfive**: LowFive is a data transfer layer that utilizes the HDF5 data model for in-situ workflows through the use of in-memory data and MPI message passing (p.e MPICH [122]). This technology enables the handling of multiple reads and writes of data from both traditional HDF5 files in physical storage and those stored in memory. LowFive also facilitates the redistribution of data from n producing processes to m consuming processes.

In Chapter 3, the CD/CV prototype is explained, which employs a parallelism technique based on containers (as depicted in figure 3.4). This technique allows the user to specify the number of threads for processing. Based on this configuration, a corresponding number of services (or clones) run as containers at every stage of the workflow, facilitating simultaneous processing of the data load.

To offer an alternative to the container approach that encapsulates the service, a do-it-yourself (DIY) technique is proposed to employ block-parallel processing by encapsulating the service within the block concept. This method allows for executing the same program/service using one or multiple threads per MPI process for every stage of the CD component's workflow in the proposed solution.

Moreover, it is suggested to incorporate the LowFive technology as a data transfer layer between the MPI processes to enable sharing or accessing

[2]`https://gitlab.kitware.com/tpeterka/diy`

the data produced by each of the processes across the various stages of the workflow (CD component).

The details of this technique are described through the application of a case study described in Section 5.4.

4.5 Summary

In this chapter, the design of new blockchain platform independent optimization techniques has been shown. New atomic transactions and grouped validation techniques were proposed, which are applied on top of the blockchain platform and that are not dependent on any specific platform. As a result, improvements were obtained in the data collection transactions protocol and the data storage procedure.

The optimizations carried out on the transactions allowed increasing the performance and scalability of transactions validation in the blockchain platform, allowing high speed validation and storing the data consistently in near real-time, allowing the data to be available for query in the continuous validation of the contract.

The ETL model has also been adapted to incorporate HDF5 technology in the CD/CV prototype to make it more suitable for HPC environments. It has been proven that it can lead to efficient and speedy processing of data at each stage of the workflow in the CD Component. Moreover, it can also minimize the number of records required in the blockchain for maintaining the audit trail of an asset, without compromising its traceability.

Finally, two technologies (DIY and Lowfive) are mentioned as a complement or alternative solution to the parallelism approach initially proposed in the CD/CV prototype.

CHAPTER 5

EXPERIMENTAL RESULTS

In the process of solving a problem, it is crucial to have a solution that is effective and meets the stated objectives. However, once a solution has been developed, it is necessary to evaluate its performance to ensure that it meets expectations. This chapter presents the methodology used to evaluate the proposed solution and explains how the different analyses were carried out to obtain the results. It is important to note that the evaluation of the solution is not only fundamental to determine its effectiveness, but also to identify opportunities for improvement and to make necessary adjustments in the future.

5.1 Evaluation methodology

An experimental evaluation was conducted in the form of case studies to investigate the data processing from IoT environments with the proposed solution. Three different scenarios were considered for this study. The first scenario involved processing user mobility information, while the second scenario involved using temperature, GPS, and acceleration data to monitor and control the routes of a fleet of trucks. The third scenario involved the analysis of electrocardiogram signals. In all three scenarios, the data handling was performed through continuous delivery processing. The research work also verified the transactions performed at each stage of these data flows.

The methodology utilized to evaluate the CD/CV prototype and the proposed optimization techniques is illustrated in Figure 5.1.

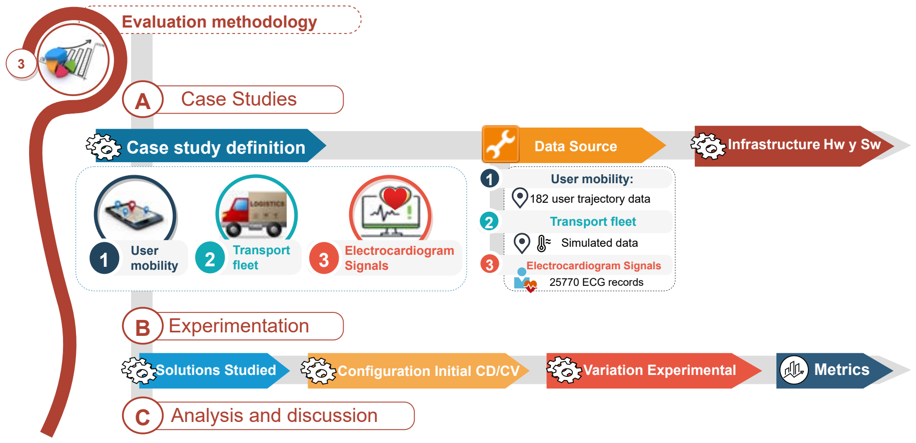


FIGURE 5.1: Evaluation methodology

The first step in this methodology is to define the use cases based on the type of IoT data and the need for process verification. Next, the hardware and software infrastructure required to implement the use cases are determined. Afterwards, the solutions to be evaluated are defined along with the parameters that will be used to experimentally vary the solutions. The workflow for processing the case study information is prepared, deployed, and operated using the CD/CV Management Framework. Finally, the metrics for the case study are established and the results are analyzed and discussed.

The following is a description of each of the case studies and the evaluation methodology used in each of them.

5.2 Case Study 1: User Mobility

For this research work, an experimental evaluation is conducted in the form of a case study based on the processing of IoT data about mobility, interest points, and trajectories of user.

This section thus describes the CD/CV schemes built for creating a traceable and verifiable workflow for the case study, which consists of user mobility data captured from GPS devices.

5.2.1 Description of case study

The mobility data are acquired and processed by software that implements the algorithm proposed by Montoliu *et al.* [123]. The algorithm allows the extraction of points of interest (POIs) by analyzing the location points of a user. The workflow includes this algorithm and implements stages such as acquisition, preprocessing, processing, storage, and exhibition.

Figure 5.2 depicts a workflow for the extraction of points of interest (POIs) from the mobility data of users.

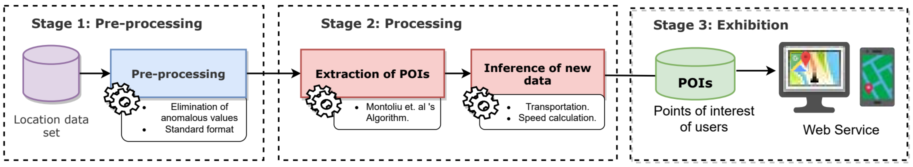


FIGURE 5.2: Design of the workflow for mobility case study.

The first stage (St_1) of the workflow prepares the mobility data extracted by the acquisition service. This stage includes data preparation procedures such as removing anomalous values, unifying formats, and transforms them into a structured form of the data, such as JSON files.

The processing stage (St_2) includes tasks for data processing, such as the extraction of POIs and the inference of new data. Given a list of location points of different users, this stage extracts points of interest (POIs) for each user. Subsequently, from the extracted POIs, relevant information about the user's mobility is inferred, such as the mode of transportation (for example walking, by bus, or by airplane) between the POIs and the mobility speed of these users.

The calculated POIs are stored in a database deployed on the third stage (St_3). Finally, an exhibition module based on a web service shows the POIs obtained per user and creates a route in a map created by GIS-geoportal. The end-users of this visualization web service can define search criteria (e.g., a given user or spatio-temporal parameters). The workflow described composes the CD/CV system displayed by the Global Manager (see Figure 5.4).

It is important to note that the signature (hash) of each extraction, transformation, and load of these sensitive data performed by each stage (in the CD scheme part) are registered in the traceability network (CV scheme part) in automatic and transparent manners.

Data Source An Acquisition service extracts records from Geolife Database [124], which contains information on the paths of 182 users collected over five years in Beijing, China. This data source includes 18,670 trajectories of an equal number of users corresponding to 1,292,951 kilometers. The acquisition service collects data such as latitude, longitude, and timestamp of each point collected by different GPS devices. The Geolife dataset is a popular dataset that has been used in many SOTA works for predicting future locations [125], multimodal locomotion with mobile devices [126], for classification of transport modes using ensembles [127], or for anonymization using machine learning [128].

Although a data file extracted from Geolife is used in the case of mobility, the system behaves as if it were an online data system, since workers deal with input data as it arrives. The data emitters are responsible for generating load dynamically.

In CD/CV the platform is dynamically deployed with the number of workers specified. The number of workers does not change based on the load, rather they are used as a default pool. Currently the system does not have the ability to dynamically adapt to the input load. It has been left for future work to have an elastic system.

5.2.2 Infrastructure Hardware and Software

The stages of the workflow were encapsulated into virtual containers using the Docker platform. In this case study, the CD/CV manager prototype was deployed on an Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz with 64 GB of memory, two hard drives of 2.7 TB each one, and 12 cores.

The virtual containers also include software such as an operating system with installation type “Compute Node” version Linux CentOS 7 x64, Cloudera Manager, Docker, and Docker Compose higher than 17.06.2-ce, cURL Latest, Go 1.11.x, Python 2.7.x, Node.js Runtime, and NPM 8.9.x, and Hyperledger Fabric 1.4.

5.2.3 Solutions Studied

A set of solutions is defined, including multiple configurations of CD/CV and a workflow engine available and studied in the literature.

The solution *DagOnStar* [10], a workflow engine for scientific and environmental processing, was studied as a related solution found in the literature.

It is also studied the solution proposed in this research (**G**lobal**M**anager: GM-CD/CV), which not only creates the workflow but also the network of verifiability. A version of the workflow only using the continuous delivery scheme (*GM-CD*) was also evaluated.

This evaluation compares the performance of GM-CD/CV with that produced by GM-CD to observe the overhead generated by the CV scheme part (registering transactions in the blockchain). A comparison between GM-CD/CV and DagOnStar was performed to measure the impact of parallelism patterns on the reduction and even elimination of CV overhead, where DagOnStar also considers implicit parallelism at the stage level.

5.2.4 Configuration Initial CD/CV

The acquisition service invokes the workflow (W) defined by the following expressions:

$W = \{St_1, St_2, St_3\}$, where: $St_1 \rightarrow$ Preprocessing stage, $St_2 \rightarrow$ Processing stage, and $St_3 \rightarrow$ Exhibition stage, $NuSt_1 \rightarrow$ N° of preprocessing services, $NuSt_2 \rightarrow$ N° of processing services, $NuSt_3 \rightarrow$ N° of exhibition services.

The CD/CV Global Manager presents the above notation in the form of boxes to the end-users to retrieve from them the ETL model and the number of services to be deployed at each stage. The Global Manager uses this information to create an architectural pattern for the efficient analysis of GPS data.

The parameters of the number of organizations ($Orgs$) and the number of peer nodes per organization (N_{peers}) for this case study are defined as follows in the continuous verifiability component (CV scheme part):

$$Orgs = \{Org_1, Org_2, Org_3\} \text{ and } PxO = N_{peers}$$

where $Orgs$ defines participating organizations, and N_{peers} determines the number of peers for each organization. In this study, $Orgs$ is equal to three, which are the number of stages in the workflow and the value of the number of peers per organization (N_{peers}) is defined in the experimental variation described in Section 5.2.6.

The result of the established definitions automatically creates a network of verifiability that is displayed below.

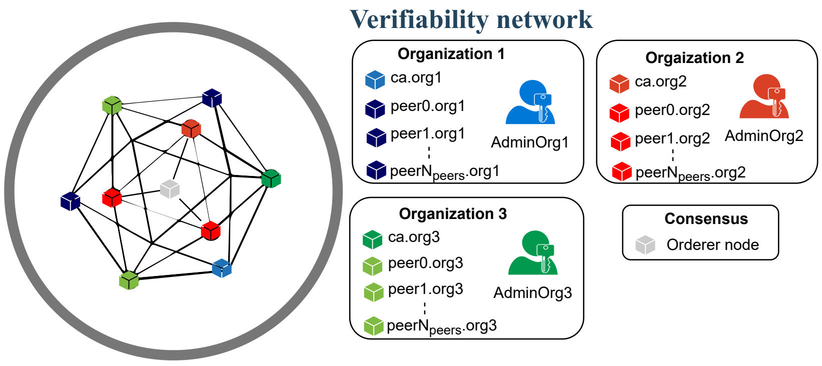


FIGURE 5.3: Verifiability network for the mobility case study.

By default, the CV scheme part is created by the *administrators* defined by each organization participating in a workflow.

When both CD (workflow) and CV (blockchain) have been established by administrators, the Orchestrators configure: the path to the Geolife database, the ETL routes for each service (pre-processing, processing, and exhibition) deployed at each stage, and the path that points to the Docker configuration for each virtual container image of the services considered. The Launchers deploy the virtual containers considered in CD/CV schemes. And the Choreographers establish the execution of the first stage of the workflow. This means that when end-users initiate the acquisition stage, the processing starts, and this culminates in the visualization, for end-users, of a map including the point of interests detected through the IoT dataflow.

At the same time, the Construction Manager with the CV scheme creates the business model and smart contracts for the stages of the workflow. As well as it creates the continuous verifiability (CV) network by using the Hyperledger Fabric. Additionally, it verifies the availability of this network for the stages of the workflow.

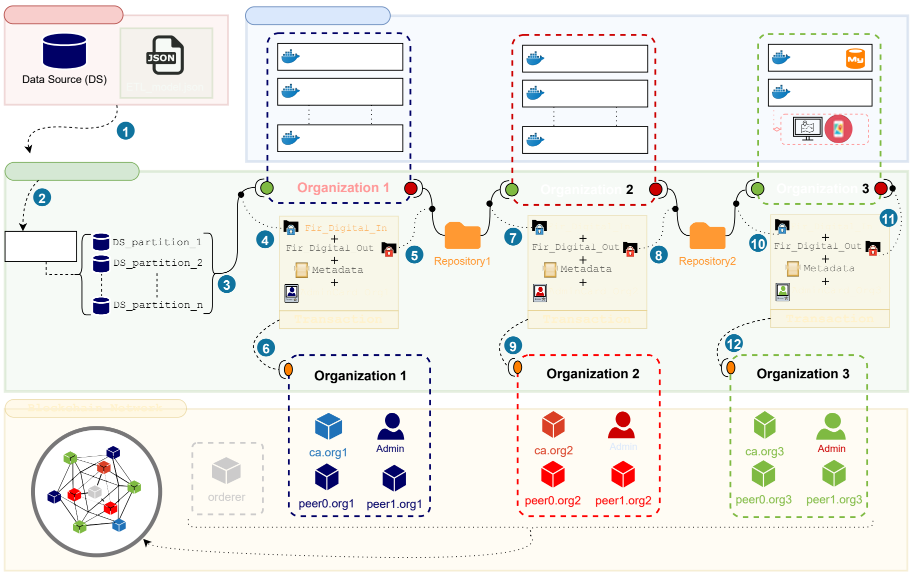


FIGURE 5.4: Operation of the CD/CV system for the user mobility case study.

5.2.5 Metrics

The following metrics were defined and captured in the execution phases (preparation, deployment, and operation) of the workflow by the three solutions studied:

1. Response Time (RT): This metric represents the time that elapses since the user made requests to the IoT dataflow until user gets a

response. For this research work, the response time is the time that the user applies a load to the workflow until user gets the results.

2. Deployment Time (DT): It represents the time required to deploy and have in service all the processing units or virtual containers. This time also includes runtime (Ru), which represents the time interval in which a deployment script runs on the operating system.

3. Overhead (Ov_d): This metric represents the extra time that a specific operation or process may require if the system adds some functionality. In this case, we seek to measure the overhead produced by adding continuous verification functionality to a traditional workflow such as DagOnStar or on our GM-CD proposal.

4. Throughput(Th): It corresponds to the volume of data per unit of time processed through the IoT dataflow.

5.2.6 Exploratory evaluation phase

To evaluate the performance of the CD/CV schemes built to conduct the previously described case study, we performed an exploratory evaluation phase to identify the impact of critical parameters on CD and CV components performance as well as a direct performance comparison of CD/CV schemes with a state-of-the-art solution (DagOnStar [10]).

In the exploratory experimental evaluation, the parameters of both the continuous delivery (CD) and continuous verifiability (CV) components are systematically varied.

The parameter variation is defined to explore the performance of the CD/CV schemes for different workloads. This allowed us to determine the set of parameters of the proposed solution that produce the best performance.

This phase considers a heterogeneous distribution of data for the number of mobility routes of each user.

Experimental variation - Continuous Delivery

The following parameters were varied in the exploratory phase of the Continuous Delivery scheme part: *a)* number of services per phase or parallelism {1,2,4,6,8,10,12,24}; *b)* volume of data to be processed $\{1, 10, 100\}$ records; and *c)* seed of the random number generator $\{1, 2\}$.

The Operation Manager selects a subset of data or users using two different seeds (each seed value generates a different subset of data). Considering all the possible combinations of the parameters (experimental *Variety* and *Volume*), there are a total of 24 configurations to measure the performance of the IoT dataflow using the CD scheme of GM-CD solution, with the same subset of data selected for each variation of data D_i for each S_i service. We ensured that each solution processes the same subset of source data. This subset of data is selected by setting a seed value for the random number generator and allows for replication of the experiments.

Two experiments were performed to apply different workloads (a subset of data) to the workflow, E_1 and E_2 (with seed 1 and 2 respectively), each of them with 24 configurations, having a total of 48 evaluations.

For the exploratory phase, the parameters were systematically varied and those that allowed to obtain the best performance in a few iterations (approximately 5) were selected for a second comparative phase with a state-of-the-art solution. Subsequently, in the comparative phase, each experiment was run 31 times. The results presented in the comparative phase show the average and standard deviation of the 31 runs (see tables 5.1 and 5.2). In most cases, the confidence interval is 5% and even smaller when the data load is lower (p.e. 10 users).

Experimental variation - Continuous Verifiability

For the CV scheme part, we considered varying the number of *Peers* in the blockchain for each organization: {1,2,4,6}. In this sense, for the GM-CD/CV solution, the total of configurations to be evaluated is given by the parameters of *Variety*, *Volume*, *Seed* (defined in the continuous delivery scheme), and the variety of Peers per organization.

This means, considering these four parameters, we performed 96 configurations of GM-CD/CV solution with the same workload for all data variations (D_1, D_{10}, and D_{100}) for each seed. In these configurations, D_i represents the amount (i) of data or users to be processed in the IoT dataflow.

In a similar fashion used for GM-CD configurations, we performed two experiments for GM-CD/CV, E_1 and E_2, each of them with the 96 configurations using the same seeds (1,2) selected for the GM-CD solution to process the same data but this time considering records in a verifiability network.

The result is a total of 192 evaluations to evaluate the GM-CD/CV solution, considering the component of continuous delivery and continuous verifiability in the exploratory stage of the experimental variation. In the comparative stage with the DagOnStar solution, we adjusted the parameters for GM-CD and GM-CD/CV according to the results presented in the exploratory phase. Section 5.2.8 describes this comparative stage.

5.2.7 Results and discussion of the studied solutions

The first analysis carried out in the exploratory phase consisted of the number of virtual CD/CV containers to be deployed for processing the mobility data for the CD scheme part (*services per stage*) and CV (*peer nodes per organization*).

Figure 5.5 shows the number of different types of containers deployed per each notation S_i-P_j, where i is the number of services per stage, and j is the number of peers per organization of a given configuration defined in the exploratory phase.

For this mobility case study, the CD containers that correspond to the workflow are *Database, Exhibition, Preprocessing Stage* and *Processing Stage.* The CV containers in the verifiability network are *Ca Nodes, Order Nodes, Client Nodes, Peer Nodes,* and *dev-Peer Nodes.* The container CD/CV Manager is the solution proposed in this work of research.

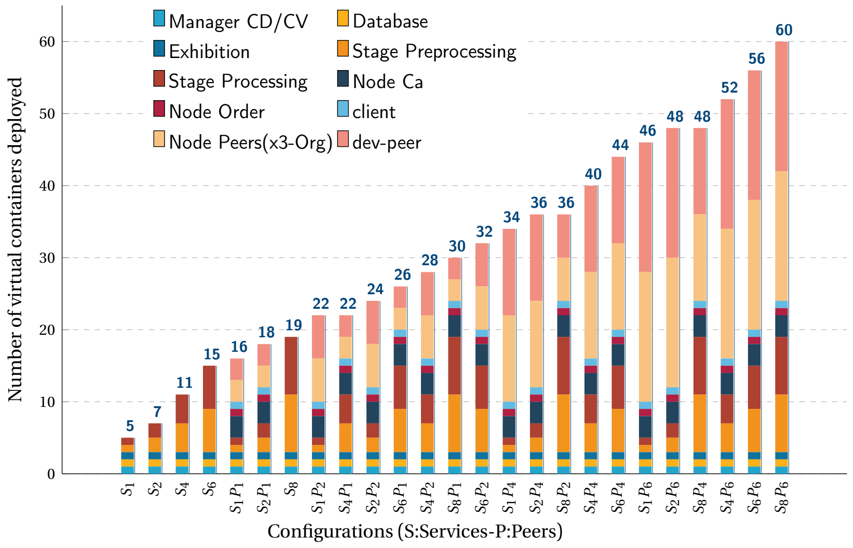
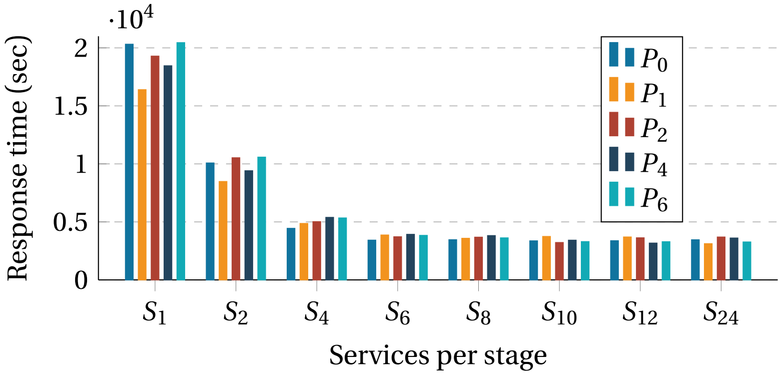


FIGURE 5.5: The number of CD/CV Virtual Containers deployed for each configuration for the mobility scenario - Exploratory stage.

As an example, Figure 5.5 shows that the notation S_8P_6 deploys 60 virtual containers on the cloud. In this example, eight correspond to the preprocessing stage and eight to the processing stage, given the notation S_8. For practical purposes for the development of this research work, all the configurations included one virtual container for the database and another one for the exhibition module. These 18 containers mentioned above are included in the CD scheme (for continuous delivery in the workflow). In the same example of S_8P_6 notation, the P_6 indicates that each organization (three for this case study) has deployed 6 peer nodes in the verifiability network, (i.e., the three organizations have deployed 18 virtual containers implementing peer nodes of a private blockchain). As previously established, the verifiability network includes two more containers for each organization: the first one implementing the order manager node and the last one implementing the certification authority (CA) node, which results in 6 virtual containers. A virtual container called *client* is deployed by each organization to perform operations such as creating the channel through which the peer nodes communicate, joining the peer nodes to this channel, among others. Also, for each Peer Node deployed, the Manager deploys another virtual container (dev-peer) to configure the business network in the Peer Node. In this sense, for the verifiability network, there would be 41 virtual containers CV. The last virtual container of the S_8P_6 configuration is the *CD/CV Manager*, which includes Orchestrator, Launcher, and Choreograph. It is important to note that both CD and CV virtual containers are automatically deployed by *CD/CV Manager* on the cloud and interconnected by using *CD/CV* scheme without the intervention of the end-users, which only provides the CD/CV manager with ETL information and business logic model (based on a DAG).

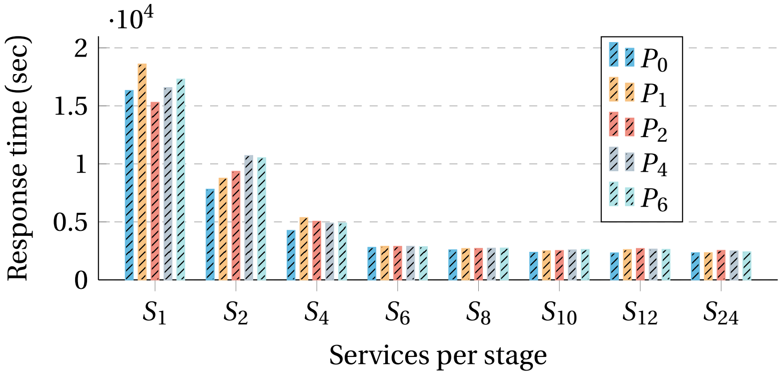
The deployment of each virtual container, either CD or CV on a given infrastructure, implies the consumption of memory, disk space, and other computational resources. Figure 5.5 shows that, as expected, those configurations with a higher number of peers (e.g., P_4 and P_6) consume more resources than those using few peers. In this context, an appropriate approach is to vary the number of peers depending on the infrastructure, analyzing its memory resources, computational power, among others. The variation of this parameter is described in Section 5.2.6.

The notation P_0 is equivalent to the solution *GM-CD* as it does not consider peer nodes per organization. The configurations with the parameters P_1, P_2, P_4 and P_6 basically are used in *GM-CD/CV* configuration by varying the number of peer nodes per organization.



Workload of 100 selected user data chosen by **seed 1**.

(A)



Workload of 100 selected user data chosen by **seed 2**.

(B)

FIGURE 5.6: Response times of the configurations in the exploratory stage.

Figures 5.6a and 5.6b show that, independently of the workload or the number of peer nodes per organization, the configurations with the parameters

S_{10}, S_{12} and S_{24} produce the shortest response times in comparison with the other variations. Even from the S_6 configuration, a stabilization of the proposed solution with the used infrastructure is achieved.

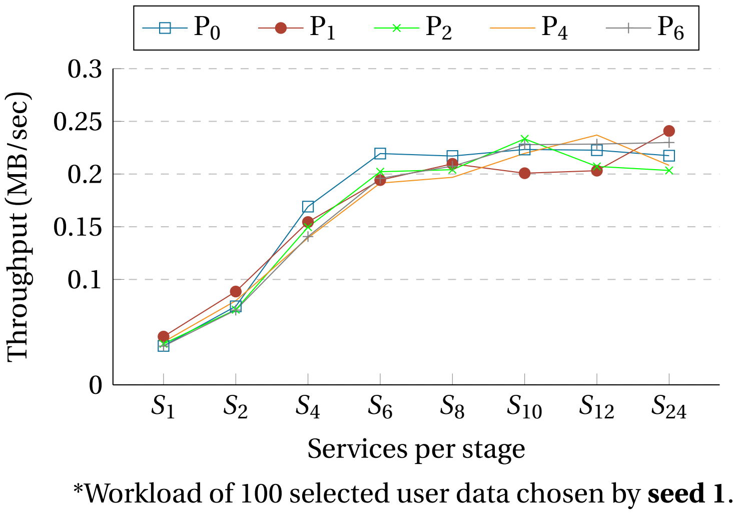
The performance when varying the number of peer nodes per organization (P_1, P_2, P_4, and P_6) could be affected by latency in the case of organizations selecting multiple cloud resources to create their verifiability network.

Figures 5.7a and 5.7b show the throughput of each *GM-CD* and *GM-CD/CV* configurations processing data of 100 selected user trajectories with seed 1 and 2 respectively.

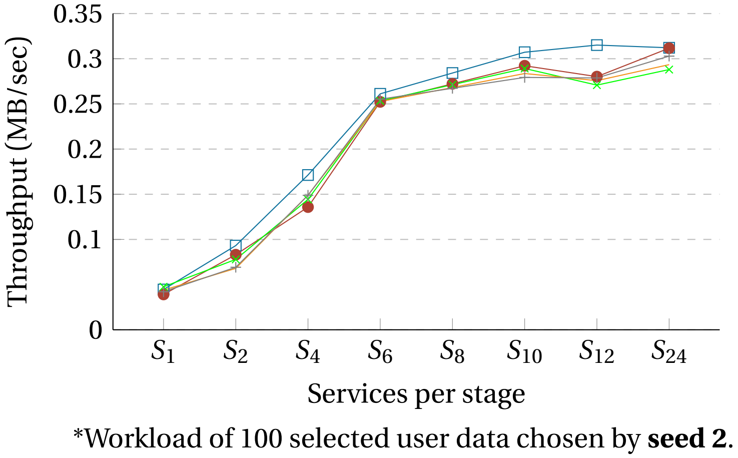
As it can be seen in the Figures 5.7a and 5.7b, the higher performance is achieved when parallelism schemes are applied to the CD/CV schemes (see configurations S_2 to S_{24}) as these configurations process data in concurrent manner using an implicit parallelism model embedded into the CD virtual containers, which also includes a load balancing algorithm (see P_1,P_2,...,P_6 when each stage using four services). This model compensates and even eliminates the costs of the CV components (See S_{2-6} for all P_1,P_2,...,P_6 CV components), which making feasible to create a traceable and verifiable critical workflows.

The results presented in this exploratory phase were analyzed to determine the parameters producing the best performance. In this context, it was observed that the choosing of the number of services per stage should be defined according to the number of cores available in the infrastructure where workflows based on CD/CV schemes were deployed on.

Specifically, in the infrastructure used in this exploratory phase, 4 and 6 peer nodes for each organization resulted in an overuse of resources (i.e., memory and disk) for CD/CV schemes, which directly affected the response time. Although the configuration of one peer node per organization produced acceptable response times, it might not be adequate as the organizations



(A)



(B)

FIGURE 5.7: Throughput of configurations defined in the exploratory stage.

would be unable to access the network records of verifiability (blockchain) in events of node failures.

5.2.8 Performance comparison.

With the results obtained in the first phase, adjustments were made to the given parameters to carry out, in a second phase, the direct performance comparison between a state-of-the-art solution (DagOnStar) and the solutions proposed in this research work (GM-CD, GM-CD/CV).

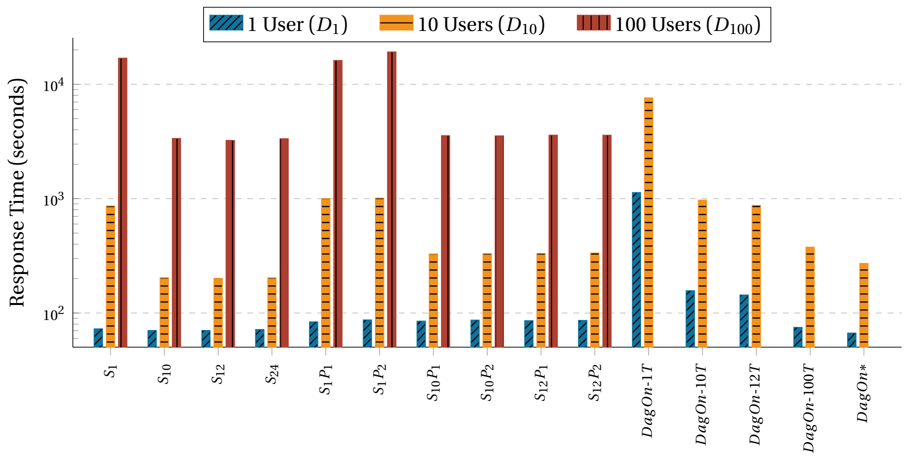
During this phase, the different solutions are configured in a comparable way (e.g., processing threads), the different response times are determined applying different workloads to both sequential and parallel versions. Additionally, the overhead/gain of a given solution is analyzed with respect to the remaining configurations.

Results and discussion of the solutions studied

The results of the exploratory phase allowed us to identify criteria to chose CD/CV operation parameters. For the continuous delivery component, the parameters to consider for the comparative stage were: 10, 12 and 24 services per stage (S_{10}, S_{12}, and S_{24}) with three different workloads ($D = 1, 10, 100$). Additionally, we included $S = 1$ to consider a sequential solution, generating a total of 12 configurations to evaluate.

Analyzing parameter criteria for GM-CD solution

Figure 5.8 shows that S_{12} was the best selection of the number of services per stage for the mobility scenario workflow in GM-CD configuration as it processed the highest load in the shortest time. In a similar way to the exploratory phase, the best response times were obtained by the configurations using as many services as available cores in the infrastructure (12 cores). When configurations deploy more threads than available cores (e.g., S_{24}), the system ending up creating a queuing of tasks in the available cores



GM-CD (S_1,S_{10},S_{12},S_{24})
GM-CD/CV (S_1P_1,S_1P_2,$S_{10}P_1$, $S_{10}P_2$,$S_{12}P_1$,$S_{12}P_2$)
DagOnStar ($DagOnStar$-1T, $DagOnStar$-10T,$DagOnStar$-12T, $DagOnStar$-100T,$DagOn*$).

FIGURE 5.8: Response time all solutions.

of the infrastructure, which reduces the improvement in response times when deploying more services than available cores.

Analyzing parameter criteria for GM-CD/CV solution

For the CV components, the number of peer nodes $P = 1,2$, produced the best performance results in the exploratory phase. We discarded the S_{24} configurations, as deploying more concurrent services than available cores (12 cores) does not produce performance improvements (-3.46% on average of all evaluations between S_{12} and S_{24}).

It was also observed that, from 10 services, the response times tend to become shorter and configurations do not yield a substantial improvement (see performance of configurations as S_{1-10}).

Figure 5.8 shows that regardless of the number of peers selected (P_1 or P_2), configurations with S_{10} with a load of 100 data (D_{100}) produces similar average response times (0.32%).

Analyzing parameter criteria for DagOnStar solution

DagOnStar also performs task parallelism using threads. For the experimentation, we considered the number of threads $T = \{1, 10, 12, 100, *\}$ with the same workloads already evaluated (data of users $D = \{1, 10, 100\}$). The parameter $T = *$ means that DagOnStar manages all the available resources of the infrastructure, creating a thread for each input task to be processed. For this purpose, DagOnStar incorporates a component known as SLURM [106], which is a cluster manager for scaling up thousands of processors.

Figure 5.8 shows the response times obtained when considering a sequential version of DagOnStar ($DagOn$-$1T$) and how it improves with the number of threads enabled for processing ($DagOn$-$10T$, $DagOn$-$12T$, $DagOn$-$100T$). The DagOnStar configuration that allows using all the available resources ($DagOn*$) with the SLURM component was the one that obtained the best results, reducing by 96.43 % the response times for the sequential version of DagOnStar for a load of 10 data and 94.09 % for an of 1 data.

The DagOn* solution only performs continuous delivery (CD) tasks and does not include the verifiability component (blockchain network) as the solution proposed in this thesis. This means the performance of DagOnStar is similar to GM-CD performance. The comparative results between these solutions are described in the next section.

The DagOnStar solution only shows results for one and ten workloads (D_1 and D_{10}) because the configurations processing 100 user data was could not be evaluated because of memory issues (threads consuming tens of Gigabytes). In particular, in the creation process of dependencies between tasks, DagOnStar creates ten sub-tasks for each data to be processed. Moreover, this configuration executes each sub-task to ensure continuous data delivery in the workflow. This process resulted in more than 10,000 tasks that the solution loaded into memory during runtime workflow, which producing overflows when DagOnStar processing workloads of 100 users' data in the infrastructure used for this experimental evaluation.

Comparative analysis and discussion of the solutions studied

The overhead produced by GM-CD/CV, and GM-CD (sequential version) configurations is evaluated and discussed in this section. The DagOnStar sequential version presents a high overhead (10x for 1 data and 7x for 10 data) compared to GM-CD sequential configuration.

Figure 5.9 shows that GM-CD/CV configurations compared to GM-CD generate mean overheads between 12% and 35%. Additionally, there will be a higher overhead when considering a higher number of peer nodes and the workload increases.

In other words, the higher number of peer nodes in the verifiability network, the longer the time to accepting a transaction in the consensus protocol of the blockchain, which directly impacts the response times perceived by the user in the workflow.

To evaluate the efficacy of the parallelism features of the solutions to handle these overheads produced by the verification component in the workflow performance, we evaluated the parallel solutions of GM-CD/CV and DagOnStar with 10 (S_{10}) and 12 (S_{12}) threads each, which were the best configurations of both solutions.

The four configurations of GM-CD/CV ($S_{10}P_1$,$S_{10}P_2$,$S_{12}P_1$ and $S_{12}P_2$) for 1 user have an overhead between 16.6% and 18.51% compared to the sequential solution GM-CD, while DagOnStar with 10 and 12 threads limit has an overhead between 97.43% and 115.39% with respect to GM-CD. These overhead percentages occur because there is only 1 User to be processed, and multiple threads are becoming unusable. However, when the load increases (D_{10} and D_{100}), the four GM-CD/CV configurations previously mentioned, reduce the overhead between 61.22% and 61.97% for a 10-data load and between 75.07% and 75.39% for a 100-data load with respect to GM-CD, while DagOnStar with 10 threads still has an overhead of 12.39% and 1.24% with 12 threads for 10 data.

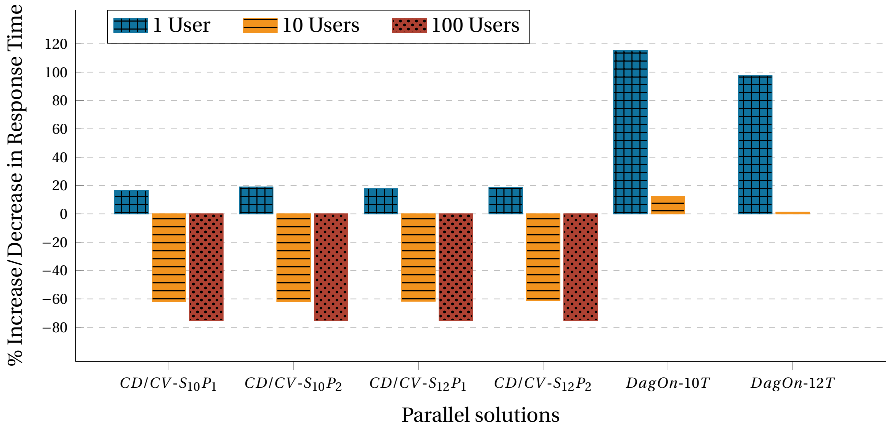


FIGURE 5.9: Comparison of the performance overhead/gain of the studied solutions with $CD\text{-}S_1$ configuration.

Figure 5.9 shows that the GM-CD/CV configurations including either 1 or 2 peer nodes with 10 and 12 threads do not produce overhead compared to the traditional sequential GM-CD workflow for 10 and 100 user workloads (D_{10} and D_{100}). The overhead (19.24%) of CV components is constant, but it is only reduced when using parallel patterns. The performance of the workflow (CD/CV) is even improved when using 10 and 12 concurrent services over GM-CD was approximately 61% for 10 data and 75% for 100 data.

When DagOnStar applies 10 and 12 threads, it still has a considerable overhead compared to the sequential GM-CD solution. However, the percentage was considerably reduced to 1.24% with 10 data when using 12 threads.

To find a competitive DagOnStar configuration, it was allowed to DagOnStar to launch as many threads as defined by SLURM (see DagOn*). Figure 5.10 shows the response times of each of the studied solutions considering 10 and 12 threads and DagOn*. As it can be seen, DagOn* can significantly improve the performance of the workflow, but it also imposes a high consumption of computational resources producing task queuing, which also producing overflows when processing a large number of concurrent workloads (e.g., DagOn* produced an overflow when processing mobility routes of 100 users).

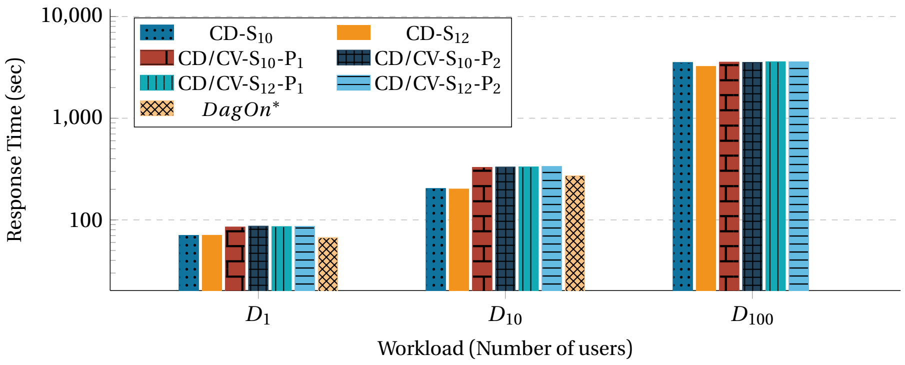


FIGURE 5.10: Response times of the studied solutions considering task parallelism.

As it can be seen, for the 10-user-data workload, the GM-CD solutions with 10 and 12 threads obtained the best response times as these configurations do not to deal with the blockchain transactions registration overhead.

As expected, the GM-CD/CV configurations (S_{10}-P_1, S_{10}-P_2, S_{12}-P_1, and S_{12}-P_2) produced overhead in comparison with parallel configurations (GM-CD and DagOnStar), which are not registering transactions. Nevertheless, this overhead is reduced when managing high-workloads as the CV can consolidate the registering of transactions, which reduces the overhead produced by CV components in comparison with parallel GM-CD solutions. For instance, the overhead of GM-CD/CV solution was with the S_{10}-P_2

parameters (10 services per stage and two peer nodes per organization) was 8.80% compared to the best GM-CD solution, which was with 12 processing threads.

The transaction recording is based on the input and output batch of each of the processing stages of the workflow and not by the user's paths. This coarse-grained strategy allows the execution times of the transactions in the verifiability network not to be dependent on the number of trajectories of a user and to be practically constant, while for the continuous delivery component, the more paths to process, the longer the response time of the processing of these tasks. In this sense, we estimate that the higher the workload evaluated in the proposed GM-CD/CV solution, the lower the cost overhead of using a verifiable network compared to the GM-CD solution applying the same load.

We recall that DagOn* achieves better response times than configurations with the verifiability component (CV) for a 10-data load because DagOn* only performs continuous delivery (CD) tasks and not making registration in the blockchain.

The average response times and percentage overhead of all configurations of the studied configurations processing 10-data workload (D_{10}) is compared with the best DagOnStar configuration in Figure 5.11.

As it can be seen, the $CD/CV\text{-}S_{12}P_2$ is the configuration producing less overhead in comparison with the parallel solution, which is still acceptable in comparison with the best performance configuration.

Table 5.1 shows a summary of the average response times produced by the studied configurations. It also includes deviation and the deviation % of the metrics for a 10-data load. The deviation percentage of the values obtained for the mean performance average of CD configurations in the

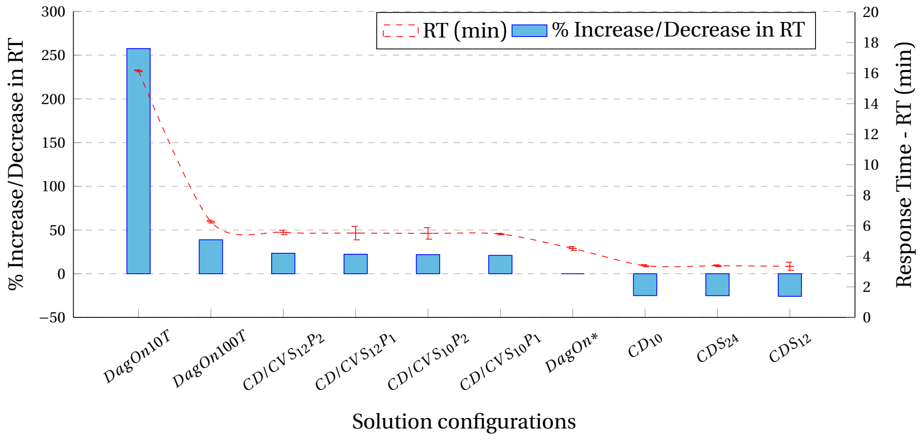


FIGURE 5.11: Overhead/gain percentage of all configurations in comparison with DagOn* for workload of 10 user data.

range of [1.31 and 2.66%], [1.29 and 3.63%] for CD/CV, but reducing when increasing the number of services.

	GM-CD				GM-CD/CV						DagOnStar			
	S_1	S_{10}	S_{12}	S_{24}	S_1-P_1	S_{10}-P_1	S_{12}-P_1	S_1-P_2	S_{10}-P_2	S_{12}-P_2	T_{10}	T_{12}	T_{100}	$T*$
Average	862.4	203.2	201.0	203.0	1001.7	327.9	331.6	1010.7	330.3	334.4	969.3	873.1	376.3	271.1
Standard deviation	22.91	3.76	2.64	3.34	36.39	11.58	7.54	23.76	5.33	4.32	8.47	37.68	26.81	22.58
% Deviation	2.66	1.85	1.31	1.65	3.63	3.53	2.27	2.35	1.61	1.29	0.87	4.32	7.12	8.33

TABLE 5.1: A summary of the mean response times produced by the studied configurations with 10-user-data workload.

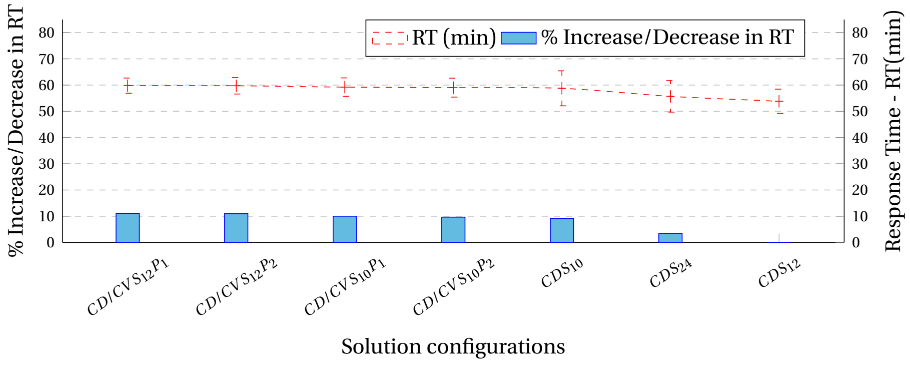


FIGURE 5.12: Overhead percentage comparison of studied configurations with CD-S_{12} for 100 user data.

The average response times and percentage overhead of all configurations of the studied configurations processing 10-data workload (D_{10}) is compared with the best CD configuration in Figure 5.12.

As it can be seen, the CD/CV-$S_{12}P_2$ is again the configuration producing less overhead in comparison with the parallel configuration producing the best performance (under 10%). It is important to note that the more the workload to be processed, the more the reduction of the overhead produced by the CV components. The reason for this behavior is due to the coarse and fine-grained strategies of GM-CD/CV. First, the coarse-grained method generates a record to the blockchain for each batch of incoming and outgoing information from each service, this is suitable for scenarios where data generation is constant. As a result, the costs of continuous verifiability are reduced by logging a consolidation of data transactions (a report based on the data flow), which reduces the number of logging processes in the verifiability network. Additionally, the fine-grained strategy is only incorporated into each ETL transformation process (for each transformed file) to generate a digital signature for each piece of data arriving and leaving each stage of the workflows.

Table 5.2 shows a summary of the mean response times produced by the studied configurations in comparison. It also includes deviation and the deviation % of the metrics for a 100-data user workload. The deviation percentage of the values obtained for the mean performance in the solutions that include the CV model varies in the range of [3.37 and 6.11%].

	GM-CD				**GM-CD/CV**						**DagOnStar**			
	S_1	S_{10}	S_{12}	S_{24}	S_1-P_1	S_{10}-P_1	S_{12}-P_1	S_1-P_2	S_{10}-P_2	S_{12}-P_2	T_{10}	T_{12}	T_{100}	$T*$
Average	16903.8	3357.3	3229.9	3341.7	16340.7	3553.2	3587.3	19196.9	3541.7	3584.3	!	!	!	!
Standard deviation	2929.2	231.2	277.1	360.2	551.3	210.7	173.6	1095.5	216.4	188.6	!	!	!	!
% Deviation	17.33	6.89	8.58	10.78	3.37	5.93	4.84	5.71	6.11	5.26	!	!	!	!

TABLE 5.2: A summary of the mean response times produced by the studied configurations with 100-user-data workload.

Analysis of the verifiability component of the best solution GM-CD/CV

We analyze the best solution of GM-CD/CV to determine the construction times of the verifiability network. Taking the results previously described into account, we consider that the best configuration found for the GM-CD/CV including two peer nodes with ten services per stage (CD/CV-S_{10}-P_2).

Although the results between CD/CV-S_{10}-P_2 and CD/CV-S_{10}-P_1 are very close, it is advisable to have a larger number of peer nodes per organization for an eventual failure of a peer node. For example, in the configuration of 1 peer node (CD/CV-S_{10}-P_1), a breakdown in the communication with the only peer node of an organization could be critical since it could disable the access of that organization to the verifiability network and not be able to issue records to the network.

Figure 5.13 shows the execution times of the processes executed in a sequential and phase-dependent manner when deploying our proposed solution.

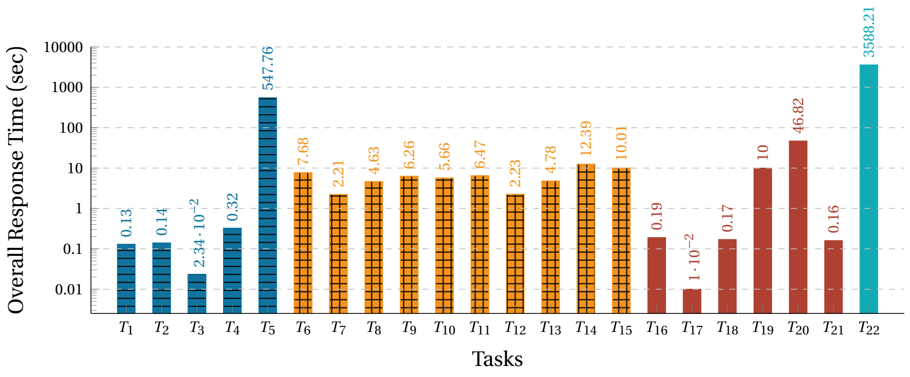


FIGURE 5.13: GM-CD/CV module runtimes.

Figure 5.13 shows the runtime of the different GM-CD/CV modules. T_1: creates configuration files; T_2: creates certificates; T_3: Inserts keys in configuration files; T_4: creates channel artifacts; T_5: Deploying CV Containers;

T_6: Building connection profiles; T_7: Creating Verifiability Network Admin Cards; T_8: Importing Admin Cards; T_9: Installing BNA on HF; T_{10}: Obtaining Admin identities; T_{11}: Starting Business Network in Network Fabric; T_{12}: Creating Administrators per organization; T_{13}: Importing Administrator Cards; T_{14}: Verifying to Verifiability Network; T_{15}: Creating REST API; T_{16}: Configuring ETL Model; T_{17}: ETL Model Hash; T_{18}: Loading ETL Model; T_{19}: Deploy CD Containers; T_{20}: Service Registration (Participants); T_{21}: Task Assignment; and T_{22}: Overall Response Time.

The first seven tasks (T_{16}:T_{22}) illustrate the execution and service times of each process of phases such as declaration, deployment, and operation of the management of CD/CV components. The tasks of the preparation phase (T_{16}, T_{17}, T_{18}) correspond to the time to create the configuration files necessary to ensure continuous delivery (CD) in the workflow from the notation DAG given by the user by using the ETL model. Subsequently, in the deployment phase, the proposed solution executes the task T_{19}, which consists of the deployment or start-up of the services of each of the stages, which we call CD containers. When the solution correctly deployed the services on the infrastructure used in the experimentation, the manager registers these containers as participants of the verifiability network (T_{20}) and assigns them the workload to be executed by each one (T_{21}). Finally, the manager launches the workflow in the operation phase, where it is calculated the response time (T_{22}) of processing the 100 data user workloads through the workflow.

We divided into two phases the processes executed for the construction, deployment, and operation of the verifiability network: the first corresponds to the infrastructure of the verification network based on the creation of the blockchain, and the second corresponds to the business network that models the logic of the transactions that the solution recorded in the blockchain. We describe the times used in each of these phases below:

1. Continuous Verification Phase 1 - Creation and deployment of the verifiability network. In this phase, the verifiability network is built, deployed, and put into production employing the Hyperledger Fabric module. The times obtained in each of the processes performed in this phase correspond to tasks T_1, T_2, T_3, T_4, and T_5 shown in Figure 5.13.

2. Continuous Verification Phase 2 - Business Network: In this phase, the business network is built, deployed, and installed in the verifiability network employing the Composer module. The times obtained in each of the processes performed in this phase correspond from task T_6 to task T_{15}, which we present in Figure 5.13.

Among the results, the best configuration obtained in the comparative phase of the user mobility case study shows that the time required to deploy the CV component (tasks T_1 to T_5 of the Hyperledger Fabric component and T_6 to T_{15} for the Business module) is much greater than that required by the CD component (tasks T_{16} to T_{21}) to perform continuous delivery. A scenario that considers a higher number of peer nodes per organization than the one described by the best configuration will generate more deployment overhead for the CV component than the CD component. The process identified as T_{22} includes the processing time of the entire workload and the verifiability network registration times that contain asset registration and transaction registration.

5.2.9 Discussion

We conducted a case study focused on the analysis of user mobility data, which revealed that GM-CD/CV not only registered the transactions in IoT dataflow, but its parallel patterns also reduced the overhead produced by the verifiability processes. A direct comparison with state-of-the-art solutions that produces continuous delivery (without a verifiability network) supports this claim.

GM-CD/CV is able to manage records in the verifiability network with coarse and fine grain strategies. The experimental evaluation showed that the coarse-grained method, applied to information batches, is suitable for scenarios where data generation is constant (sensors). On the other hand, the fine-grain strategy is used in each ETL transformation process (for each transformed file) to generate a digital signature for each data arriving and departing to/from each stage of the workflows.

The evaluation revealed that the coarse-grained strategy reduces the costs of continuous verifiability by recording a consolidation of data transactions (a dataflow-based report), which reduces the number of registration processes in the verifiability network. This strategy is more suitable for IoT dataflows producing data at constant rates (data produced by sensors in time series). In turn, the second strategy registers, in real-time, each transaction performed during a dataflow (stage-based report) in the blockchain, which however increases the costs of information recording. This strategy is more suitable for large data produced asynchronously (e.g., when managing files not streaming).

In performance details, the experimental evaluation also revealed that the number of peer nodes in the CV model deployed for each organization significantly impacts the performance of the CD model. Two peer nodes per organization were sufficient to perform transaction recording with service availability without imposing relevant costs on the CD model.

The experimental comparison of GM-CD/CV with a state-of-the-art solution based on parallel stages (DagOnStar), revealed that DagOnStar does not allow highly concurrent processing of workloads (it was not possible to process trajectories of 100 users in the mobility scenario) while GM-CD/CV did not show this limitation. DagOnStar is quite competitive when processing short tasks and is not limited to launching execution threads for stages (e.g., processing ten users). The best configuration of DagOnStar produced a 20% performance improvement compared to GM-CD/CV, but knowing that DagOnStar does not use the verifiability network (not performing blockchain records). In scenarios where GM-CD/CV and DagOnStar used the same number of resources, the overhead was not only eliminated, but even GM-CD/CV produced a better performance than DagOnStar (82.88% on average for 10 and 12 threads configurations with 1 and 10 user loads).

Our contributions are important for those companies and organizations requiring traceability of the processes executed along the several applications composing a workflow. Moreover, the continuous verifiability allows firing automatic processes to generate alarms, maintenance actions, or corrective processes to solve failures in the data-flow processing, which is important for supply chains, health workflows, business, and many other sectors. Thus, our solution has many potential applications, and applying it to several sectors with real-world problems will be part of our next steps.

5.3 Case Study 2: Fleet of trucks transporting food

The case study presented below (transport of goods with trucks including three sensors) is used to evaluate the optimizations proposed in chapter 4 of this research work. Mainly, the techniques to be applied are focused on managing the data produced from the IoT environment directly on the blockchain. Subsequently, the periodic verification process described in chapter 3 is evaluated.

5.3.1 Description of case study

The case study developed consists of the business logic of food transport. In this business logic different organizations participate, each organization has its fleet of trucks, each truck makes different shipments of assets and incorporates three types of sensors: temperature, GPS, and speed to control the conditions of each shipment. The scheme of the deployment is shown in Figure 5.14.

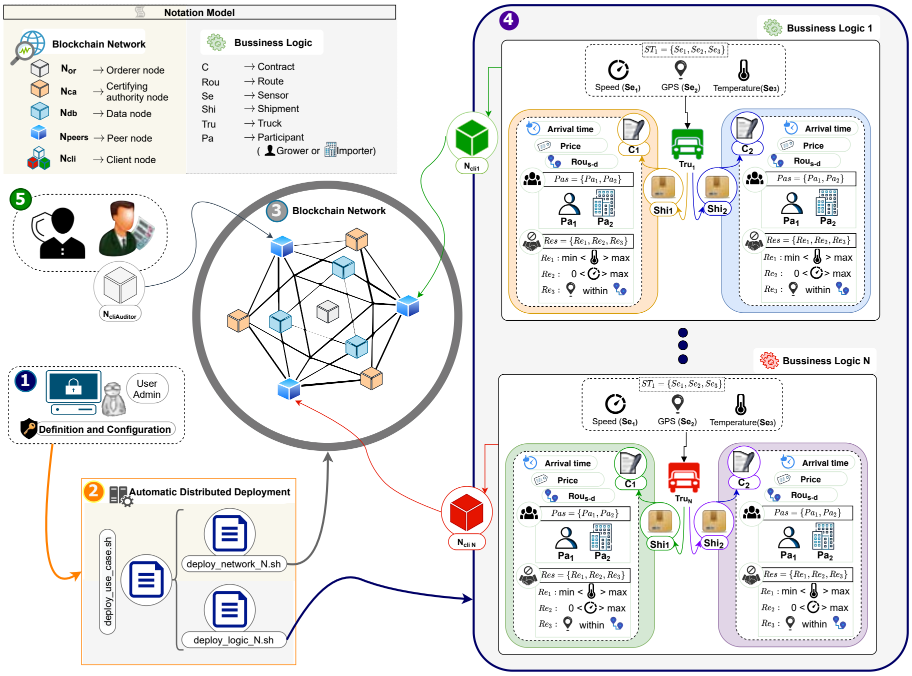


FIGURE 5.14: Case study Transport fleet

Data source

For this case study the data were generated synthetically for each type of sensor. Below, an example of IoT data generation is presented using the transportation use case.

Let´s assume that each Tru_i truck collects data with their respective sensors $ST_i = \{Se_1, Se_2, ..., Se_n\}$ where i corresponds to the truck's identifier and n is the number of sensors it may have installed. In turn, each sensor has associated a *frequency rate (fr_{se_i})* in seconds to determine the frequency of data collection. For example, the Se_1 sensor has an associated frequency rate fr_{se_1}, if the value of fr_{se_1} is 60 seconds, it means that every minute the Se_1 sensor belonging to ST_i of the Tru_i truck is obtaining data from its environment. The frequency rates denoted by $FR_i = \{fr_{se_1}, fr_{se_2}, ..., fr_{se_n}\}$ may vary from each other, regardless of whether they belong to the same Tru_i truck.

If the sensors' data have to be logged in the blockchain infrastructure, a transaction (Tx_{se}) has to be performed to record the data collected for each sensor, following the frequency rate of each ST_i sensor, in the blockchain verifiability network. Each transaction is denoted as $Tx_{se} = \{Se_{id}, Shi_{id}, timestamp, value\}$, where $<Se_{id}>$ is the sensor identifier, $<Shi_{id}>$ is the shipment identifier, $<timestamp>$ is the data capture date, and $<value>$ is the set of values collected by the sensor. In this context, the number of transactions to the blockchain, issued by a shipment ($Tx_{shi_i\_truck_j}$) is given by:

$$Tx_{shi_i\_truck_{j-k}} = \sum_{i=1}^{n} fr_{se_i} \tag{5.1}$$

Where i corresponds to the number of shipment made by the truck j of the fleet k, and n corresponds to the total number of sensors installed on the truck j.

Therefore, the number of blockchain transactions that are made is given by:

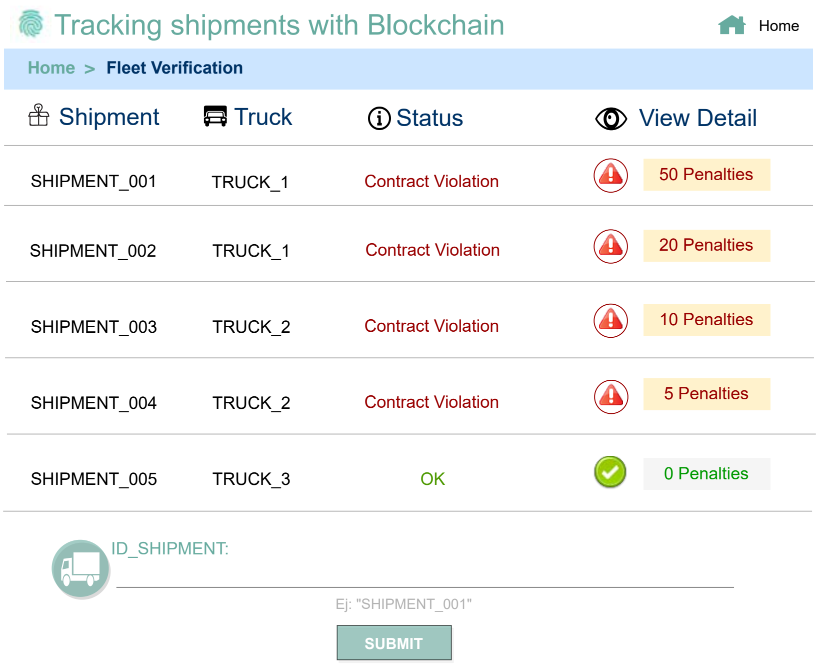
$$\#\_Tx\_data\_collected = \sum_{k=1}^{nf}(\sum_{j=1}^{nt}(\sum_{i=1}^{ns} Tx_{shi_i\_truck_{j-k}})) \tag{5.2}$$

Where nf corresponds to the number of fleets, nt is the number of trucks, ns is the number of shipments, i corresponds to the id of shipment made by the truck j of the fleet k. As may bee seen, the amount of data increase geometrically with number of devices and the frequency (samples per second) of each device.

T particularize the example, we define three sensors per truck: temperature, GPS, and speed. Moreover, we assume that sensors may send data with a different frequency. In our use case, the temperature sensor sends a measurement every second, while the GPS sensor sends the truck's location every ten seconds, and the speed sensor every five seconds, allowing to know in real-time the status of the foods and the truck, to make decisions quickly, as will be explained in the following lines.

Verification process

The verification process consists of determining the status of the shipments made by each truck by verifying whether the records of temperature, speed, and GPS sensors of each truck meet the conditions established in the contract. In particular, for the temperature recording is verified that it is within the minimum and maximum values established in the contract, for the speed sensor that does not exceed the maximum limit allowed and for each GPS recording is verified that this point is within a minimum distance to any of the GPS points that make up the route that the truck must follow. The verification process is done incrementally, from checking a shipment to checking the status of a fleet of trucks. As an example, it is possible to check the general state of a truck, that is to say, to check the state of all the shipments that a particular truck has made. For this purpose, the verification history is reused, since there is a record with the verification results of each shipment.

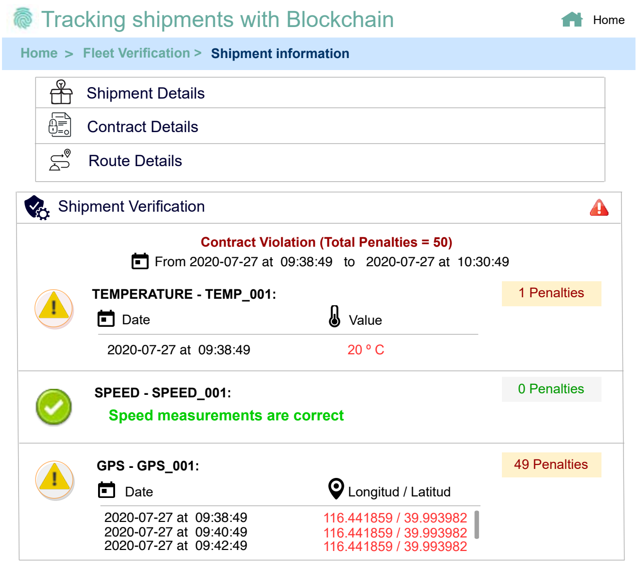


Shipment	Truck	Status	View Detail
SHIPMENT_001	TRUCK_1	Contract Violation	50 Penalties
SHIPMENT_002	TRUCK_1	Contract Violation	20 Penalties
SHIPMENT_003	TRUCK_2	Contract Violation	10 Penalties
SHIPMENT_004	TRUCK_2	Contract Violation	5 Penalties
SHIPMENT_005	TRUCK_3	OK	0 Penalties

FIGURE 5.15: Search for validated shipments

Figure 5.15 shows an output example of contracts tracking for a fleet of trucks. The list indicates the shipments made by each truck and the contract status of that shipment (Ok or Contract Violation). In the case of contract violation, the number of penalties that have been recorded during the complete verification of the shipment is initially shown.

Each shipment can then be examined in more detail if needed just by clicking on the Penalties icon. The proposed system allows the authorized actors to view the details of the shipment and the contract, the route that the truck should follow, and the details of the result of verification (Figure 5.16). The verification process generates as a result all the data captured by the sensors that do not meet the conditions of the contract. As an example, Figure 5.16 shows that, in a failed shipment, there was a violation of the contract for temperature data and a GPS location outside the established route.


Tracking shipments with Blockchain Home

Home > Fleet Verification > Shipment information

Shipment Details

Contract Details

Route Details

Shipment Verification

Contract Violation (Total Penalties = 50)

From 2020-07-27 at 09:38:49 to 2020-07-27 at 10:30:49

TEMPERATURE - TEMP_001: 1 Penalties

Date	Value
2020-07-27 at 09:38:49	20 º C

SPEED - SPEED_001: 0 Penalties

Speed measurements are correct

GPS - GPS_001: 49 Penalties

Date	Longitud / Latitud
2020-07-27 at 09:38:49	116.441859 / 39.993982
2020-07-27 at 09:40:49	116.441859 / 39.993982
2020-07-27 at 09:42:49	116.441859 / 39.993982

FIGURE 5.16: Detail of a validated shipment

Thus, our web tool allows the participants of the blockchain network, the government entities, or auditors in charge of supervising the correct execution of the contracts in business logic, to easily consult the data of the blockchain network and the verification results of any business logic, even if our use case was developed for fleets of trucks transporting food.

5.3.2 Infrastructure Hardware and Software

As our goal was to implement an efficient closed network, the Hyperledger Fabric platform was chosen because it provides better performance [110].

We used the version 1.4.2 of this tool and the Kafka protocol [129]. As for the hardware used to deploy the network, a general-purpose computer has been used for each organization in the network, which allowed us to see the performance when using general-purpose computers and not large data centers.

5.3.3 Solutions Studied

The goal of this case study is to investigate how well traditional and proposed techniques perform in terms of recording blockchain transactions. To achieve this, we will focus on analyzing the solution suggested in this research (Global Manager: GM-CD/CV). This solution not only establishes the workflow but also the verifiability network, where the proposed techniques are analyzed.

5.3.4 Configuration Initial CD/CV

The blockchain network deployed to cope with all this data follows the architecture shown in Figure 5.17.

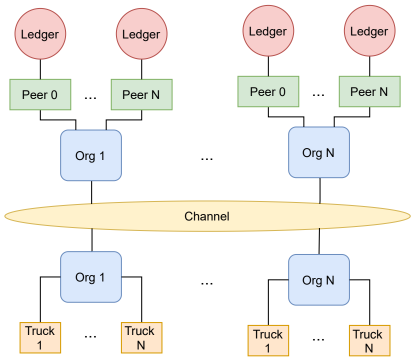


FIGURE 5.17: Generic blockchain network design

For the purpose of evaluation, we composed a blockchain scenario with three different organizations connected by a single channel. The transactions registered in one of the organizations are replicated and stored in the other organizations within the network. Each organization is composed of two peer nodes that are responsible for the transaction validation and storage in their corresponding ledger. Each organization have three clients, which in this use case will be trucks that have three sensors (Temperature, GPS, and Speed). The trucks, which must belong to the fleet of a specific organization, execute the transactions to send the sensors' measurements to the network.

We define the functionalities necessary to carry out the business logic (queries and definition of functions) in the smart contract.

Even if the definition of the business logic is specific of each business case, we here describe the model with our logistic of truck shipments. Below we show the participants (company), the fleet of trucks of each company, the routes of trucks, the contracts, and the shipments that will be registered as transactions in the blockchain verifiability network. The definition of each of the entities and elements mentioned above is done in the following order:

1. **Participant creation**. A participant *Pa* is defined a tuple of values, such as <*id, type, email, address, quantity*>. The <*id*> allows the unique identification of the participant (in our case a unique name), the <*type*>identifies the role in the business (in our case can be grower, importer, or truck); <*email*> and <*address*> attributes are used to contact the participant.

2. **Fleet of trucks creation**. A fleet *Fle* is established as a set of trucks for transporting assets. The fleet is denoted as $Fle = \{tru_1, tru_2, ..., tru_n\}$, where each tru_i is a transport truck including a tuple of <*id,*

sensors> values. The sensors of each truck are denoted as $ST_t = \{Se_1, Se_2, ..., Se_n\}$, where t is the truck identifier and each Se_i is a sensor identifier incorporated in the truck, which can be temperature, GPS, speed, etc.

3. **Route creation → *Rou***. A route is established as a set of GPS points that a certain truck (Tru_i) must follow. The route is denoted as $Rou_{s-d} = \{poi_1, poi_2, ..., poi_n\}$ where s is the origin point and d the destination point, and where each poi_i is a GPS point formed by the tuple of values <*latitude, longitude, date, time*>. The <*latitude*> and <*longitude*> attribute indicate the position of the truck, and the <*date*> and <*time*> attributes are set to allow time and space restrictions on the routes that trucks must follow.

4. **Contract configuration**. A contract C establishes the parameters agreed upon by the different participants for the execution and fulfilment of a transaction. A contract is denoted as $C = \{Pas, arrival-time, price, penalty, Res\}$ where $Pas = \{Pa_1, Pa_2, ..., Pa_n\}$ is the set of participants that establish the contract, <*arrival-time*> is the time limit established for the fulfillment of the same, <*price*> is the value of the contract, <*penalty*> is the amount to pay if the contract is violated, and $Res = \{Re_1, Re_2, ...Re_n\}$ denotes the set of constraints to which the contract is subject. For example, the value of Re_1 could correspond to a minimum value of temperature that can receive an asset or Re_n could be the maximum value of the acceptable speed of the truck that transports the asset.

5. **Shipping configuration → *Shi***. Shipments are made by trucks (*Tru*) and are tied to contracts (C) and established routes (*Rou*). A shipment is denoted as Shi = {id, asset, status,unit count, C_{id}, tru_{fi}, Rou_{id}}, where <*id*> is the shipment identifier; <*asset*> is the goods or object being transferred; shipment <*status*> can be: *created, in transit*

or delivered; <*unit count*> determines the number of assets being transported; C_{id} corresponds to the contract identifier to which the shipment is tied; <tru_{fi}> identifies the *truck i* of the *fleet f* that makes the shipment; and Rou_{s-d} is the route identifier that the shipment must follow.

6. **Establish queries**. The model is designed to consult the current and historical values of the records of each of the elements or entities belonging to the business logic.

5.3.5 Experimental Variation

The following parameters were varied for this case study:

1. Number transactions: To evaluate the performance provided by the blockchain network, we measured the transaction processing average time in five different workload scenarios: 1.000, 5.000, 10.000, 20.000 and 40.000 transactions respectively.
2. MaxMessageCount: To measure the effects of executing transactions concurrently, this parameter was modified with the following values: 1, 2, 3, 5, 15, 25 and 100.

5.3.6 Metrics

The following metrics were defined and captured in the operation phase of the proposed GM:CD/CV prototype for this case study:

1. Response Time (RT): this metric represents the time from when the user issues a transaction in the blockchain network until that transaction is consolidated or confirmed in the ledger and replicated by all

nodes in the network. The sensors are emulated by having actors that produce the data of a specific sensor every T time periodically. Thus, for the near RT issue, data are similar to those of a real-world system. The transactions in the blockchain are made in a real platform, not a simulated one. Of course, using the data generator we could have created many more scenarios to test, but for this thesis those considered more significant have been defined.

2. Number of errors (NE): This metric represents the number of transactions that failed to register on the blockchain.

3. Failure Rate (FR): This represents the percentage of transactions that were not correctly recorded in the blockchain.

4. Delay Time (DT): Represents the delay time it takes for a transaction to be validated.

5.3.7 Analysis and discussion

In this section, traditional techniques implemented in the blockchain are evaluated along with two of the optimization techniques proposed in Section 4.4.

Evaluation of traditional blockchain implementation - Performance Analysis

We made a performance evaluation of the blockchain network using the concatenated records solution when it is subjected to an intense workload, but without concurrence. In other words, only one sensor of one shipment from those created in the blockchain network will send data. A temperature sensor per truck, performing a transaction every 0.5 seconds, is used to send a large amount of data to stress the system. As shown in Figure 5.18, the average execution time of a transaction increases as the total number of executed transactions raises. Therefore, it can be concluded that this implementation offers low scalability because transaction time increases quickly. The reason for this behavior is that the block size increases for every new transaction, as new data are stored concatenated with the previous values for that sensor, which means a longer time to replicate the transaction in all the peer nodes of the network.

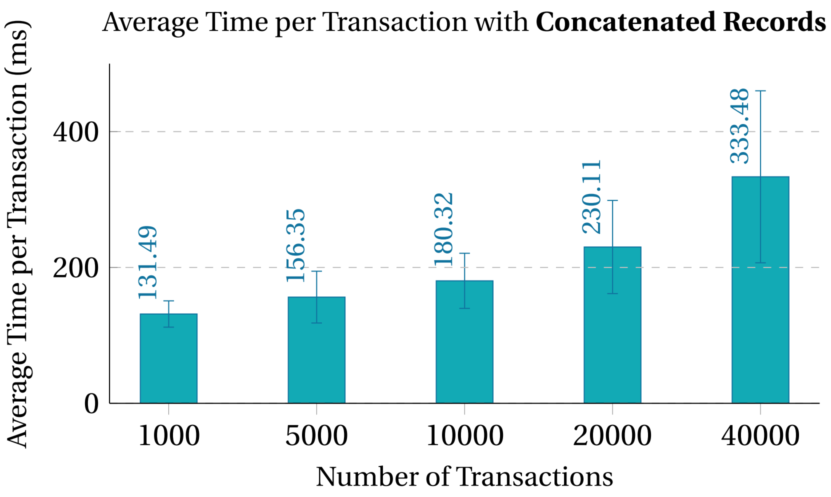


FIGURE 5.18: Average time per transaction with concatenated records with different workloads

Number of transactions	**1000**	**5000**	**10000**	**20000**	**40000**
Average Time	131.49	156.35	180.32	230.11	333.48
Standard deviation	19.37	38.12	40.56	68.56	126.59
Sensing time (minutes)	8.34	41.7	83.4	166.7	333.34

In this test, the data capture simulation is performed with a single sensor (temperature) and the sensing time varies to obtain the desired number of transactions: 1000,..., 40000. In the Figure 5.18, it can be observed that the average time to register 1 transaction out of 1000 (sensing time 8 minutes approximately) transactions is 131.49 ms with a deviation of 19.37. Additionally, it is observed that when the number of transactions is increased, under a concatenated records approach, the average time to record 1 transaction increases considerably (e.g. 333.48 ms per transaction out of a total of 40000 transactions).

Evaluation of traditional blockchain implementation - Concurrency Analysis

In addition to measure performance, we studied the behavior of the blockchain network when several sensors from trucks send data at the same time, creating concurrent transactions. The purpose of this experiment is to determine the number of errors that occur when transactions are executed concurrently and, furthermore, to determine the percentage of data that has not been recorded in the ledger despite the transactions having taken place.

For this reason, the network scenario shown in the Figure 5.17 has been used to perform this analysis, particularized with a total of nine trucks in the network with their three corresponding sensors, so that in this study there are twenty-seven sensors carrying out transactions at the same time, which ensures the existence of concurrency. To study the number of errors that happen in execution and the volume of data that is not recorded, a test of one-hour duration was executed, with the sensors sending information periodically with the following rates: temperature sensors 1 second, GPS sensors 10 seconds, and speed sensors 5 seconds.

We use the MaxMessageCount parameter (described in 11) to measure the effects of this concurrency feature, following the experimental variation described in 5.3.5.

Figure 5.19 shows the errors produced during the validation of transactions using concatenated records. This number is very high with the frequencies defined in our experimental study, specially for individual evaluation. It also shows that when the size of the block of transactions is increased, the number of errors that appear during the transaction execution decreases and the performance is increased. This behavior is due to the effect of validating multiple transactions simultaneously. The trade-off is that, as we increase *MaxMessageCount*, we also decrease the data availability for the peers. For example, if a GPS sensor is sent every 10 seconds and the parameter *MaxMessageCount* is set to 50, those position would be only available in the ledgers with an average delay of 250 seconds.

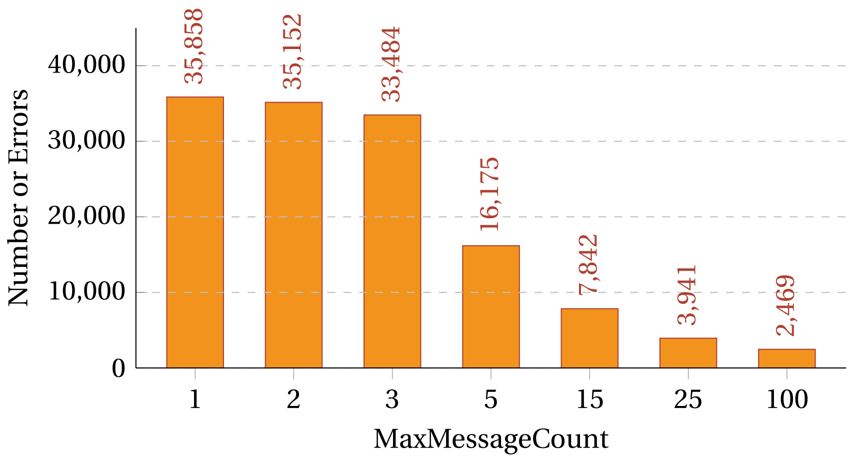


FIGURE 5.19: Execution errors using concatenated records with different validation block sizes

In this evaluation, given that there are nine clients (trucks) with three sensors each (gps, speed, temperature), data are being generated concurrently for one hour. The error obtained in the concatenated records approach is due to the fact that a transaction in the blockchain cannot be recorded until the block has been validated. Figure 5.19 shows that the highest number of errors (unrecorded transactions) occurs when only one transaction is received to validate the block. In other words, when many validation processes are applied, transactions that are not received due to system saturation are lost. On the contrary, the more transactions are received before validating the entire block, the fewer the number of errors produced.

After studying how many errors are produced during execution, the percentage of data that have not been recorded in the tests described in the previous lines was analyzed. For this purpose, the average value of unrecorded data will be calculated for each type of sensor and each test.

The results obtained in these tests are shown in Figure 5.20, where it can be seen that the losses are worst for high rate sensors, as majority of the unrecorded data comes from the temperature sensors where the failure rates are almost always greater than 50%. However, when analyzing the results obtained in the GPS and speed sensors, the failure rate is lower and, in some cases, there is no data loss.

This behavior is caused by the method of storing the data received from the sensors because the data is stored in a concatenated format and the previous measurements must be obtained first in order to add the last measurement. If the frequency of data sent is high, the previous transactions may not have been validated yet and, as a consequence, they are not available for querying because they are not stored in the ledger, which implies that the last data are not available and the data that were read between the last transaction stored in the ledger and the last one carried out are lost. Moreover, as previously mentioned, the majority of the losses are produced in the temperature sensors, as these sensors have a higher sampling frequency (1 Hz.) than other sensors, while the GPS sensors, with frequency of 0.1 Hz., show less transactions unrecorded. The failure rate is high as each new measure is added to the block in the blockchain (case MaxMessageCount = 1). That means that the block has to be collected, the new data has to be added at the end, and finally the new block has to be processed in the chain. As the data grows, the block is larger and the failure rate increase. A possible solution is to group several measures before inserting them in the block. Then the failure rate decrease, but the verification is delayed, as the contracts must wait to have the last pack of data. The problem of the block

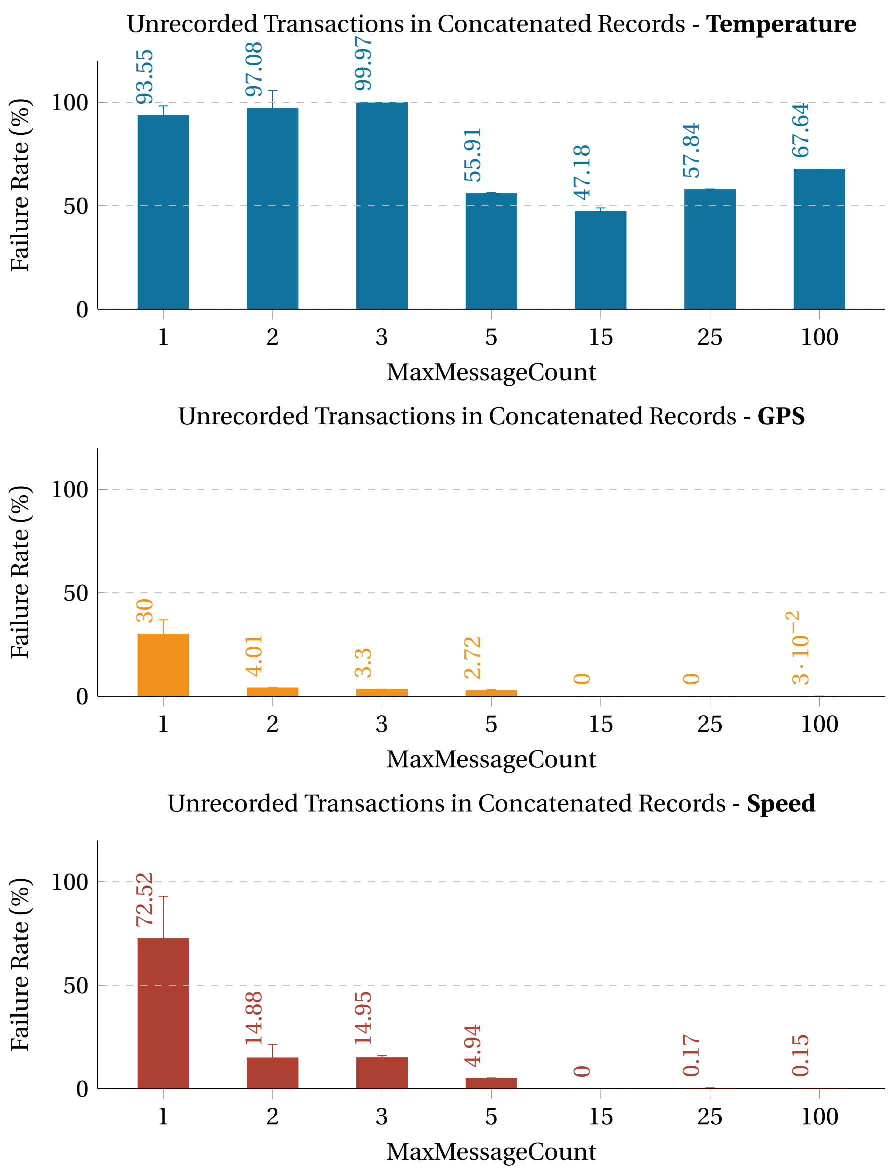


FIGURE 5.20: Unrecorded temperature, GPS and speed sensor transactions with different validation block size

size, however, cannot be avoided with this technique. Those results show that the traditional blockchain platforms are unable to record high speed transactions due to protocol latency and data storage costs.

The next section evaluates the first two proposed generic transaction processing optimizations that are compatible with any blockchain platform used to implement the blockchain network: atomic records and changing the validation block size.

Evaluation of proposed optimizations in blockchain - Atomic Records

To evaluate the correctness and performance of this optimization, we run the same experiments devised to analyze the performance of the concatenated model (Section 5.3.7). The results shown in Figure 5.21 prove that the proposed optimization provides very good performance and scalability of the blockchain network as the number of transaction increase, since the average time taken to execute a transaction is kept constant and lower than in the concatenated case. This effect is especially clear in the load test that carries out 40,000 transactions, where the time is reduced to less than half. Furthermore, it should be noted that as the execution time remains constant regardless of the total number of transactions executed, the scalability of the network is enhanced.

In contrast with the traditional blockchain, this optimization does not originate data losses, even making immediate validation of each transactions, because the transactions are atomic and the previously recorded values do not have to be recovered as they are stored concatenated. However, although all the data is correctly recorded, we observed that, when the block size is too small and the data frequency high, the blockchain network is not able to validate the transactions that are generated in due time, generating delays in the data availability for queries, because a transaction is available for querying only when it has been validated.

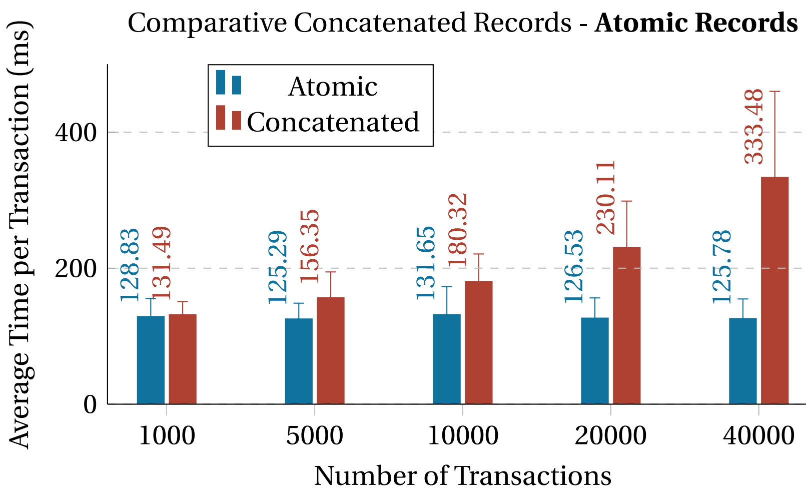


FIGURE 5.21: Comparison between concatenated records and atomic records with different workloads

Evaluation of proposed optimizations in blockchain - Changing the Transaction Block Size

Figure 5.22 shows the delays generated in validations with the different block sizes in a scenario of 27 sensors sending data for an hour. The longest delay occurs when the transactions are validated individually, which is 295 minutes. However, when transaction block sizes are increased to validate a group of 15, 25, and 100 transactions, the processing delay is not generated because the system is able to manage all the transactions that are produced on time and the transactions are stored in the ledger in a short period of time since they were created.

This optimization allows the existence of concurrency in the system and increases the performance, because, by adjusting the block size, the system is able to validate a large number of transactions in a short time, being these stored in the ledger and available for querying in due time.

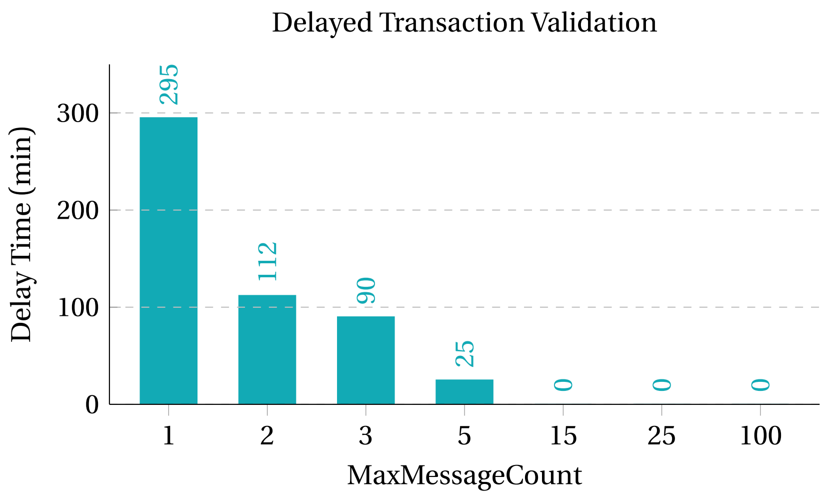


FIGURE 5.22: Delayed transaction validation with different validation block size

We run experiments to test the performance and reliability of the solution proposed. The evaluation results show that the optimizations proposed allows to process data request with higher registration rate that the solution provided in by default in blockchain platforms, like Hyperledger. We could also avoid missing data and reduce the transaction processing delays, thus increasing the reliability of the supply chain processes.

The evaluation presented in this section allowed to assess the feasibility of the blockchain optimization techniques proposed. In this case, the solution has not been compared with other techniques as, as far as I know, there are no similar solutions. As shown in the state of the art, most blockchain optimizations try to modify the platform or the distributed consensus protocol to be more efficient, and this hypothesis was not considered in this work.

5.4 Case Study 3: Electrocardiogram Signals

The objective of this case study is to evaluate the high performance delivery between workflow stages (technique proposed in Section 4.4.4 of chapter 4) of the CD component of the proposed solution. In addition, to analyze the performance of the solution in a scenario where the CD and CV components are distributed.

The primary goal of classifying electrocardiogram signals, which represent the electrical activity of the heart, is to identify any irregularities in the cardiac rhythm. Detecting such abnormalities can help issue timely alerts to prevent severe events like cardiac arrest. Therefore, it's essential to maintain an unchangeable record of the classification results obtained while processing each ECG signal.

5.4.1 Description of case study

The case study uses data from electrocardiogram signals as its basis. These signals are subjected to a series of procedures in order to be categorized by a model that has already been trained. The workflow, which is depicted in Figure 5.23, involves three primary stages: 1) collection of signals and related data, 2) segmentation of signals, and 3) classification of each individual segment of a signal. Each stage of the workflow saves the results in HDF5 format.

All the outcomes achieved during the collection of signals, segmentation of signals and the ECG signal classification process are recorded and managed by the proposed solution described in chapter 3.

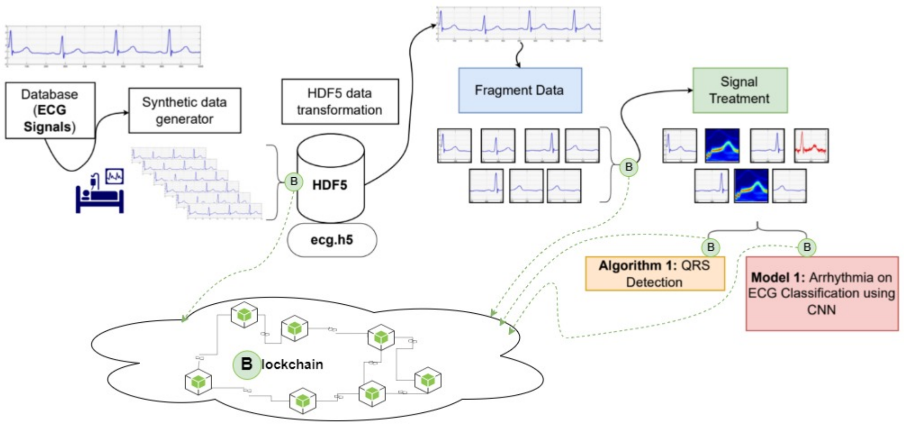


FIGURE 5.23: Case study: ECG signal processing

Data Source

We utilized a database introduced in [130] to create the case study for processing electrocardiogram signals.

The paper [130] details the creation and features of a comprehensive database of 12-lead electrocardiogram (ECG) readings with multiple labels. The database comprises 25,770 ECG readings from 24,666 patients who visited the Shandong Provincial Hospital between August 2019 to August 2020. These readings have a duration ranging from 10 to 60 seconds.

Each ECG reading is stored in an HDF5 file. That is, 25,770 files are considered as the data source for the case study described in this section.

5.4.2 Infrastructure Hardware and Software

In this case study, the prototype GM-CD/CV was deployed using the software and hardware infrastructure described in Tables 5.3 and 5.4 respectively.

As an alternative solution to the container approach that encapsulates the service/application in the workflow, this case study integrates the DIY and Lowfive technology into the software stack of the proposed CD/CV prototype. DIY [121] is a library that enables the implementation of scalable algorithms through block-parallel processing, which can be executed both inside and outside the core. It offers the flexibility to use a single program with one or more threads per MPI process, thus combining distributed memory message passing with shared memory thread parallelism in a transparent manner.

Infrastructure Software			
Description	**Version**	**Description**	**Version**
Operating System (8 nodes)	Ubuntu 20.04.5 LTS	mpi4py	3.1.4
Docker	20.10.12	zlib	1.2.13
Hyperledger Fabric	2.2.1	numpy	1.23.5
Go	1.19.4	tensorflow	2.11.0
Python	3.8.10	keras	2.11.0
gcc	9.4.0 and 11.1.0	cURL	Latest
hdf5 (parallel version)	1.12.2	DIY	2
h5py	3.7.0	Lowfive	Beta
mpich	4.0.3	Wilkins	Beta

TABLE 5.3: Software infrastructure of the case study.

For the deployment of the case study, we built a Linux cluster composed of 8 nodes with the characteristics described in Table 5.4.

5.4.3 Solutions Studied

We have established a group of solutions/designs (refer to table 5.5) specifically for evaluating this case study:

A description of each of the proposed solutions is described below:

1. **Design 1**: This proposed solution is the prototype version that doesn't incorporate the continuous verification (CV) element and operates

Server	Architecture	Processor	RAM Memory	Storage	CPU(s)	Socket(s)	Core(s) per socket	Thread(s) per core
node0	x86_64	Intel(R) Core(TM) i5-3570 CPU @ 3.40GHz	15 G	3.0 T	4	1	4	1
node1	x86_64	Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz	19 G	983 G	8	1	4	2
node2	x86_64	Intel(R) Core(TM) i5-3570 CPU @ 3.40GHz	10 G	3.0 T	4	1	4	1
node3	x86_64	Intel(R) Core(TM) i3-3240 CPU @ 3.40GHz	7.6 G	491 G	4	1	2	2
node4	x86_64	Intel(R) Core(TM) i5-3470 CPU @ 3.20GHz	7.9 G	491 G	4	1	4	1
node5	x86_64	Intel(R) Core(TM) i5-3330 CPU @ 3.00GHz	15 G	983 G	4	1	4	1
node6	x86_64	Intel(R) Core(TM) i5-3330 CPU @ 3.00GHz	7.6 G	983 G	4	1	4	1
node7	x86_64	Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz	32 G	919 G	8	1	4	2
node8	x86_64	Intel(R) Core(TM) i5-3330 CPU @ 3.00GHz	7.6 G	491 G	4	1	4	1

TABLE 5.4: Hardware infrastructure of the case study.

Id solution	Solution	Components	Software stack	Blockchain
Design 1	GM-CD: Fullstage	CD	CD ->H5Py ->HDF5 ->DIY ->MPI	N/A
Design 2	GM-CD: Greedy	CD	CD ->H5Py ->HDF5 ->DIY ->MPI	N/A
Design 3	GM-CD/CV: Greedy BC	CD + CV	CD/CV ->H5Py ->HDF5 ->DIY ->MPI	One host
Design 4	GM-CD: Greedy L5	CD	CD ->H5Py ->HDF5 ->LowFive ->DIY ->MPI	N/A
Design 5	GM-CD: Greedy L5 BC	CD + CV	CD/CV ->H5Py ->HDF5 ->LowFive ->DIY ->MPI	One host

TABLE 5.5: Solutions studied - Case study 3.

on a Fullstage scheme. The Fullstage approach processes all the data in a given stage i before moving on to stage i+1.

2. **Design 2**: This particular solution involves the proposed prototype, which doesn't factor in the continuous verification (CV) component, and operates using a Greedy scheme. The Greedy scheme processes workload immediately as it's generated, i.e, with the i+1 stage initiated as soon as any data from the i stage becomes accessible.

3. **Design 3**: This solution involves the proposed prototype utilizing a Greedy scheme, with all components of the verifiability network (blockchain) deployed on a single node.

4. **Design 4**: This solution consists of the previously described design 2 but including the Lowfive transport layer in the software stack.

5. **Design 5**: This solution involves the proposed prototype utilizing a Greedy scheme, with all components of the verifiability network (blockchain) deployed on a single node and including the Lowfive transport layer in the software stack.

5.4.4 Configuration Initial CD/CV

In this particular study, the first setup of the CD/CV model is established during the preparation stage by following the approach outlined in Section 3.2.

Three executable programs were designated for the CD component to create a workflow. Each executable program represents a stage (St) of the workflow (W) that will handle the data source (DS) as described in Section 5.4.1.

Afterwards, the ETL model was established for each of the workflow stages. In essence, this defines the specific locations where each stage (executable: T) will extract (E) the data from and where it will store (L) the resulting output.

In the case study, the Manager/Worker pattern ($Pa_{Ma/Wo}$) was employed, and it is intended to utilize the third technique suggested in Section 4.4.4 by integrating MPI. Consequently, the total data to be processed (specified later on) is distributed among N workers or MPI processes.

The configuration initial in the CV component involves determining the number of organizations (O), peer nodes (N_{peers}), and Orderer manager node (N_{or}) necessary to establish the blockchain network for recording transactions. Initially, a preliminary evaluation was conducted to define the configuration of the network. This involved considering two organizations ($O = 2$) for the entire workflow, with a single peer node ($N_{peers} = 1$) and an orderer node (N_{or}). This configuration was chosen as it represents the most fundamental setup for assessing the distributed scenario of both CD and

CV verifiability network components, which is one of the main objectives of the case study.

Lastly, the performance of the CD/CV system under a distributed scheme and utilizing the high throughput delivery technique is assessed by gradually increasing the number of organizations (one for each workflow stage) and the number of peers (with a minimum of two). This evaluation is conducted to determine how well the system functions in a more complex and distributed environment.

5.4.5 Experimental Variation

The following parameters were varied for this case study:

1. Workload (D): The parameter (D) represents the quantity of input data (hdf5 files) that will be handled in the workflow. It's essential to note that each workflow stage in this case study produces one output file for each incoming file. Hence, as there are three stages in the workflow, the total number of files processed in the workflow is equal to the value of this parameter (D) multiplied by 3. The values to be considered are $D = \{128, 512, 1024\}$.
2. Number process (P): Adopting the manager/worker pattern, this parameter represents the count of workers (MPI processes) that will operate within each workflow stage. For ease of use, the same value is applied to this parameter across all three stages of the workflow. The values to be considered are $P = \{4, 8, 16\}$.
3. Hosts (H): This parameter indicates the count of hosts or machines utilized for processing the workflow data, with the parameter values determined based on the available resources outlined in 5.4. The

values to be considered are H={1-Host-node(NFC),1-Host-node7,3-Hosts and 6-Hosts}. The setup labeled as "1-Host-node0(NFS)" means that the data processing for the workflow is happening on the same node where the network file system is mounted (node0). Therefore, the network bandwidth remains unaffected as all the files produced by the workflow are saved on the same node where the network file system is mounted.

5.4.6 Metrics

The following metrics were defined and captured in the operation phase of the proposed GM:CD/CV prototype for this case study:

1. Response Time (RT): This metric represents the time that elapses since the user made requests to the IoT dataflow until user gets a response. For this research work, the response time is the time that the user applies a load to the workflow until user gets the results. The response time result reported in the experimental evaluation corresponds to the average of 15 iterations.
2. Speedup (Sp): The speedup is a measure that compares the performance of a program or process in two different systems. It represents the number of times a program is faster in one system compared to another. The speedup is calculated as the ratio between the time needed to execute a task in an original system and the time needed to execute the same task in an improved system. The system that was used originally and the improved system are identified based on the experimental variation (see 5.4.5) and evaluation described in Section 5.4.7 of the case study.

3. Overhead (Ov_d): This metric represents the extra time that a specific operation or process may require if the system adds some functionality. In this case, we seek to measure the overhead produced by adding continuous verification functionality to a traditional workflow.

4. Throughput(Th): It corresponds to the volume of data per unit of time processed through the IoT dataflow.

5.4.7 Analysis and discussion

Exploratory stage - Scheme selection (D1: Fullstage vs D2: Greedy)

The initial evaluation involves selecting one of the two synchronization methods, namely Fullstage and Greedy, to synchronize the workflow stages.

The Fullstage method implies that the i-th stage of data flow commences processing only after the i-1th stage has completed. On the other hand, the Greedy approach notifies the i+1th stage once the workload is processed at the i-th stage, and results are acquired, indicating that the workload is ready for processing at the subsequent stage.

The results obtained with the Fullstage and Greedy approaches are presented in Tables 5.6 and 5.7 respectively.

Design 1: GM-CD (Fullstage)							
Number of process (Np):		**Np = (Stg1:4 | Stg2: 4 | Stg3: 4)**			**Np = (Stg1:8 | Stg2: 8 | Stg3: 8)**		
Data / Configuration		**6Hosts**	**3Hosts**	**1Host (node7)**	**6Hosts**	**3Hosts**	**1Host (node7)**
100	**Average**	189,94	289,20	589,27	200,39	392,58	525,67
	Stdve	3,04	6,93	6,29	7,43	11,89	3,27
1024	**Average**	1954,10	2857,80	5948,42	2020,61	3784,62	4588,49
	Stdve	29,70	86,72	23,12	31,13	51,28	9,12

TABLE 5.6: Results Design 1- GM-CD: Fullstage.

Design 2 : GM-CD (Greedy)									
Number of process (Np):		**Np = (Stg1:4 | Stg2: 4 | Stg3: 4)**				**Np = (Stg1:8 | Stg2: 8 | Stg3: 8)**			
\Configuration: Data		**6Hosts**	**3Hosts**	**1Host (node7)**	**1Host (node0=NFS)**	**6Hosts**	**3Hosts**	**1Hosts (node7)**	**1Host (node0=NFS)**
128	**Average**	**145,40**	276,42	646,97	263,25	189,35	342,02	522,71	296,24
	Stdve	1,30	5,33	0,90	4,27	5,45	8,31	1,90	4,62
	Speedup	1,81	0,95	0,41	1,00	1,56	0,87	0,57	1,00
512	**Average**	**570,98**	1081,87	2570,22	979,94	716,99	1266,03	2043,19	1131,83
	Stdve	0,85	8,86	10,44	7,84	10,49	10,24	0,63	21,13
	Speedup	1,72	0,91	0,38	1,00	1,58	0,89	0,55	1,00
1024	**Average**	**1146,07**	2158,32	5120,51	1985,38	1427,48	2517,32	4069,44	2258,67
	Stdve	3,04	20,09	10,91	55,16	14,81	20,43	2,13	17,60
	Speedup	1,73	0,92	0,39	1,00	1,58	0,90	0,56	1,00

TABLE 5.7: Results Design 2- GM-CD: Greedy.

The difference in experimental variation between design 1 and design 2 is due to the quick and clear evidence of performance seen in the initial evaluations. For instance, Table 5.6 and 5.7 demonstrate that when processing 128 files, the Greedy method performs better than the Fullstage approach, even when the latter processes a smaller data set (100 data) in almost all configurations. This implies that Greedy yields better results than Fullstage, even when dealing with a larger amount of data.

Figure 5.24 depicts the speedup attained by utilizing both synchronization methods with varying workloads, hosts and the number of MPI processes (Np) to determine the optimal performing approach.

The outcomes depicted in figure 5.24 reveal that the Greedy technique results in superior performance in the suggested system. Furthermore, the worst performance is achieved when more resources than required are assigned, i.e., when the number of processes exceeds the number of cores per host (Np=16) when varying the number of processes per stage ($Np = \{4, 8, 16\}$). This observation considers the fact that the number of available cores in the hardware infrastructure nodes (see Section 5.4), for this particular study, is either 4 or 8.

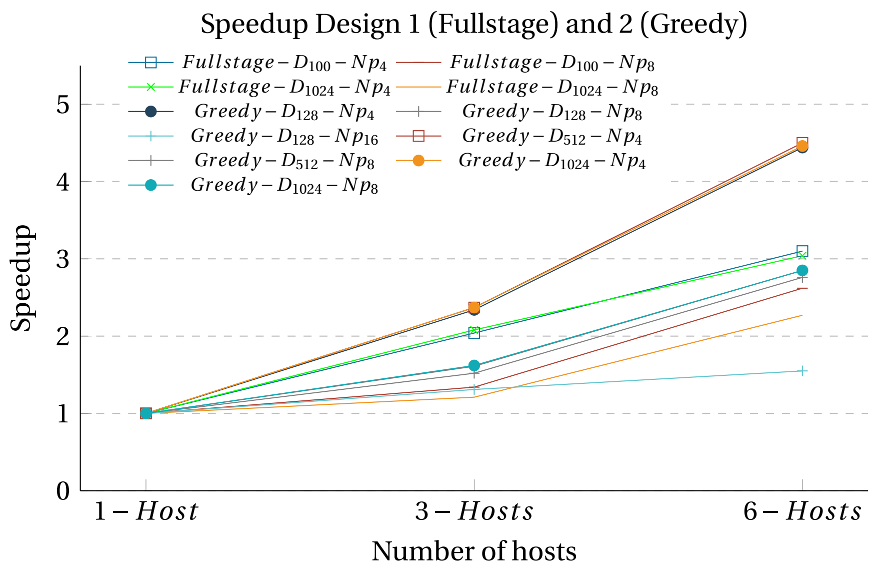


FIGURE 5.24: Speedup Design 1 (Fullstage) and 2 (Greedy).

These results allow us to conclude two things:

- A Greedy approach results better than a Fullstage approach, since it is maximizing the data flow.
- Overprovisioning (Np=16) does not pay off since the data flow models or programs in this case study consume a lot of CPU.

In this context, the Fullstage approach and overprovisioning (Np>8) are discarded for the next evaluation phases of this case study.

Solutions with continuous verification in the selected scheme.

Based on the assessment above, Design 2, which utilizes the Greedy method, outperforms Design 1, which employs the Fullstage approach. The subsequent section discusses and examines the outcomes obtained from incorporating the verifiability component (CV) into the Greedy solution, known as Design 3.

Table 5.8 summarizes the outcomes obtained from the evaluation of Design 3 in this case study.

Design 3 : GM-CD/CV (Greedy Blockchain)									
Number of process (Np):		**Np = (Stg1:4 | Stg2: 4 | Stg3: 4)**				**Np = (Stg1:8 | Stg2: 8 | Stg3: 8)**			
\Configuration: Data		**6Hosts**	**3Hosts**	**1Host (node7)**	**1Host (node0=NFS)**	**6Hosts**	**3Hosts**	**1Hosts (node7)**	**1Host (node0=NFS)**
128	**Average**	**166,54**	290,22	665,87	293,87	194,37	379,42	533,78	319,00
	Stdve	1,98	7,14	2,71	13,56	3,52	3,97	0,61	12,77
	Speedup	1,76	1,01	0,44	1,00	1,64	0,84	0,60	1,00
512	**Average**	**666,91**	1136,78	2639,03	1199,56	745,70	1398,33	2079,74	1236,24
	Stdve	5,31	16,06	2,13	8,74	10,69	15,29	1,09	26,70
	Speedup	1,80	1,06	0,45	1,00	1,66	0,88	0,59	1,00
1024	**Average**	**1352,57**	2289,26	5262,33	2563,26	1469,72	2772,95	4147,21	2536,18
	Stdve	4,37	23,40	4,57	43,94	11,69	10,05	3,49	16,70
	Speedup	1,90	1,12	0,49	1,00	1,73	0,91	0,61	1,00

TABLE 5.8: Results Design 3- GM-CD/CV: Greedy Blockchain.

Based on the results depicted in Table 5.7 and Table 5.8 for the Greedy method and Greedy with Blockchain respectively, it is apparent that the optimal performance is achieved by handling the workload using the maximum number of available hosts (6 hosts) with 4 processes assigned per stage of the workflow.

The primary reason for this is that the hardware infrastructure's nodes (as listed in Table 5.4) consist of only two nodes with 8 CPUs while all other nodes in the cluster have 4 CPUs. Consequently, the proposed solution's performance is negatively impacted in setups where the number of processes utilized surpasses the available CPUs on the host performing the assignment.

Figure 5.25 can exemplify the previous declaration, wherein Design 3 shows that every configuration (number of hosts) with 4 processes per stage ($Np = 4$) results in superior outcomes than configurations (number of hosts) with 8 processes per stage ($Np = 8$), except for one configuration involving only one host that chooses node 7 of the hardware infrastructure. In this scenario, the performance is improved by using $Np = 8$ since only node 7, which contains 8 CPUs, is utilized, whereas $Np = 4$ would lead to an underutilization of available resources.

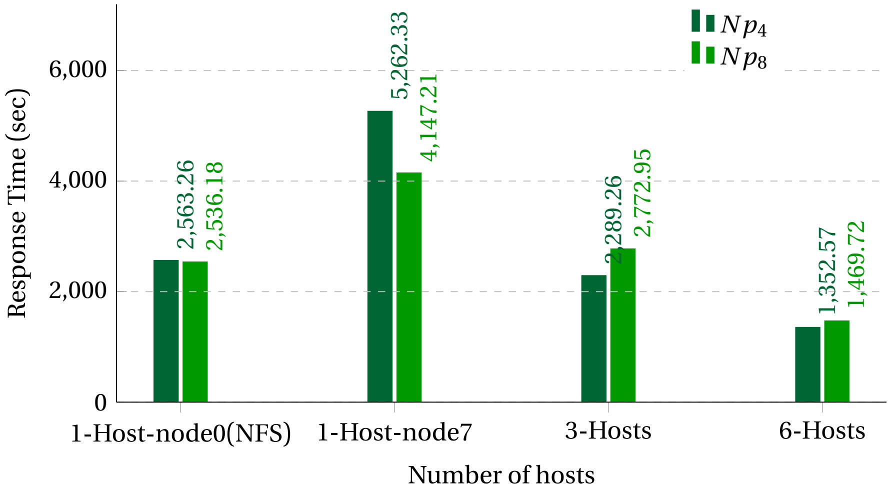


FIGURE 5.25: Comparison of the number of processes selected (Np 4 vs Np8) in design 3 to process 1024 data.

Even though the Figure in 5.25 show the results for processing a workload of 1024 data, the trend of achieving better performance only when the number of processes equals the available CPUs on the used host is also evident in the results presented for a workload of 128 data and 512 data (see Table 5.8). Furthermore, this trend is observed in the outcomes presented for Design 2, as shown in Table 5.8.

Moreover, the optimal setup for both "design 2 (Greedy)" and "design 3 (Greedy Blockchain)" is the one with 6 hosts and 4 processes, which shows that the suggested solution is practical in a distributed setting. This configuration delivers better performance than a local setup where the workflow outcomes are instantly written to disk or do not go through the network.

A comparison of this optimal configuration (6-hosts with 4 processes per stage) between design 2 and 3 is presented in the Figure 5.26.

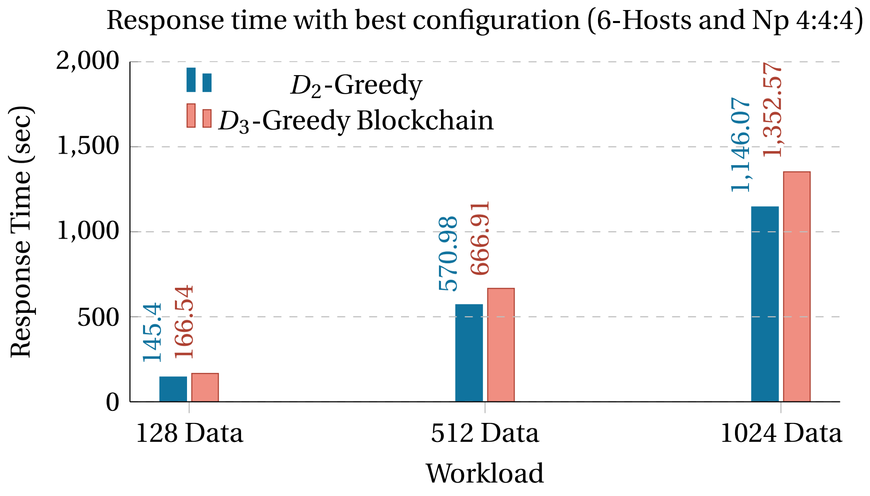


FIGURE 5.26: Comparison between the best configuration (6 Hosts and NP 4-4-4) obtained in Design 2 and 3.

Solutions with LowFive transport layer

This section assesses the practicality of incorporating the Lowfive component into the software stack of the proposed solution. To achieve this, two new designs are examined: Design 4, which involves the Lowfive component (L5) as part of the Greedy version, and Design 5, which is similar to Design 4 but also includes the verifiability component (CV).

The outcomes acquired from the fourth and fifth designs are compiled and presented in Table 5.9 and Table 5.10, respectively.

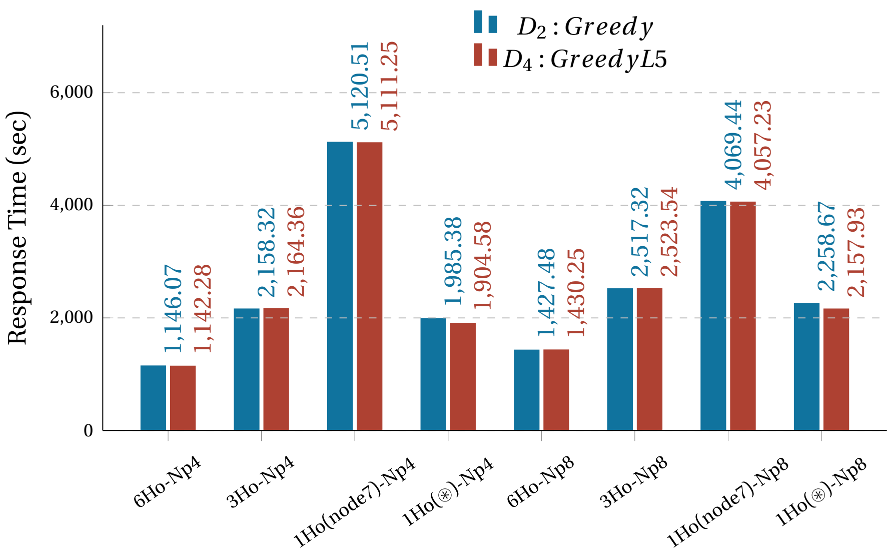
Design 4: GM-CD (Greedy L5)									
Number of process (Np):		**Np = (Stg1:4 | Stg2: 4 | Stg3: 4)**				**Np = (Stg1:8 | Stg2: 8 | Stg3: 8)**			
Data / Configuration		**6Hosts**	**3Hosts**	**1Host (node7)**	**1Host (node0=NFS)**	**6Hosts**	**3Hosts**	**1Host (node7)**	**1Host (node0=NFS)**
128	**Average**	145,27	272,85	645,74	236,50	189,51	341,45	518,81	268,94
	Stdve	1,02	11,16	2,35	1,62	2,64	9,63	1,62	1,93
	Speedup	1,63	0,87	0,37	1,00	1,42	0,79	0,52	1,00
512	**Average**	571,88	1084,83	2561,15	953,87	722,43	1276,51	2039,62	1086,89
	Stdve	2,43	11,50	11,73	3,10	6,33	10,47	10,20	3,47
	Speedup	1,67	0,88	0,37	1,00	1,50	0,85	0,53	1,00
1024	**Average**	1142,28	2164,36	5111,25	1904,58	1430,25	2523,54	4057,23	2157,93
	Stdve	9,26	37,03	16,98	9,57	8,34	17,75	5,90	7,26
	Speedup	1,67	0,88	0,37	1,00	1,51	0,86	0,53	1,00

TABLE 5.9: Results Design 4- GM-CD: Greedy L5.

The results presented in Table 5.9 indicate that in design 4 (Greedy L5) and like the previous solutions, the configuration that obtains the highest performance regardless of the workload corresponds to the deployment of the solution in six hosts and four processes per workflow stage (6H-Np4). In fact, with the 6-host distributed approach, a higher speedup is obtained with respect to the other configurations regardless of the workload or the number of processes per workflow stage.

Since in Design 4 a new component (L5: LowFive) has been added to the software stack of the proposed solution with the higher performance approach (Greedy), Figure 5.27 shows a comparison between the Greedy solution without the LowFive component (Design 2) and the Greedy solution with the addition of this component (Design 4).

Figure 5.27 shows that adding the LowFive component to the proposed solution does not represent an overhead to the overall system. In fact, the difference between the two solutions can be considered negligible considering that the range of values of these differences is between 3.79 and 12.21 seconds. These values do not represent more than 1% of the best solution in each of the configurations, except for the configurations where only node 0 is used as storage (NFS) and deployment of the entire solution, where the difference between the GreedyL5 and Greedy solution is between 80.8 and 100.74 seconds.



(**Ho**: Number of hosts /**Np** : Number of process / ⊛: NFS storage node 0)

FIGURE 5.27: Comparison of design 2 (Greedy) and design 4 (Greedy L5) to process 1024 data.

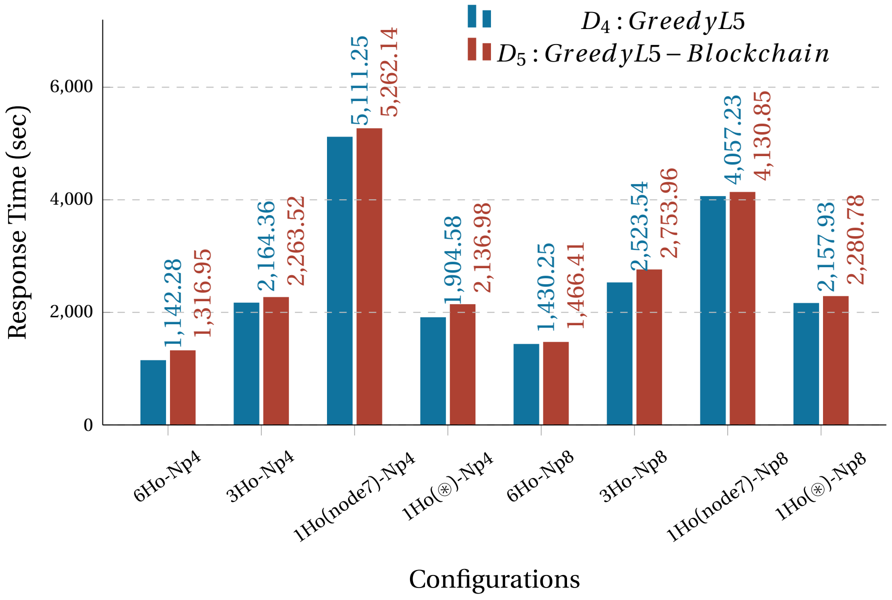
Finally, a study was carried out (Greedy L5-Blockchain) adding the CV component to the Greedy L5 solution with the objective of having a solution with the continuous verification component and with the characteristics or functionalities of efficient, distributed and parallel processing. The results of this study are presented in Table 5.10.

Design 5: GM-CD/CV (Greedy L5 Blockchain)									
Number of process (Np):		**Np = (Stg1:4 | Stg2: 4 | Stg3: 4)**				**Np = (Stg1:8 | Stg2: 8 | Stg3: 8)**			
Data / Configuration		**6Hosts**	**3Hosts**	**1Host (node7)**	**1Host (node0=NFS)**	**6Hosts**	**3Hosts**	**1Host (node7)**	**1Host (node0=NFS)**
128	**Average**	164,41	287,27	665,27	260,06	196,44	369,77	529,41	278,75
	Stdve	2,62	6,03	2,26	11,17	3,37	5,78	1,42	1,06
	Speedup	1,58	0,91	0,39	1,00	1,42	0,75	0,53	1,00
512	**Average**	653,89	1125,72	2636,09	1051,01	740,47	1377,60	2072,51	1142,67
	Stdve	2,54	13,61	6,60	5,38	10,39	14,68	5,76	4,97
	Speedup	1,61	0,93	0,40	1,00	1,54	0,83	0,55	1,00
1024	**Average**	1316,95	2263,52	5262,14	2136,98	1466,41	2753,96	4130,85	2280,78
	Stdve	4,73	38,55	6,84	6,55	7,81	12,06	1,51	7,23
	Speedup	1,62	0,94	0,41	1,00	1,56	0,83	0,55	1,00

TABLE 5.10: Results Design 5- GM-CD/CV: Greedy L5 Blockchain.

The configurations of 6 hosts with 4 and 8 processes in the workflow stages obtain the highest performance of all the evaluated configurations. As can be seen in Table 5.10, the highest system speedup is obtained with 4 processes in each of the workflow stages. All speedups are obtained from the local configuration (1Host:node0:NFS) where data are stored and processed directly and the network component does not intervene.

By adding the CV component (D_5: Greedy L5-Blockchain) to the solution with higher performance and functionality (D_4: Greedy L5), it is estimated that there is an overhead produced by the use of the blockchain network and the transactions to be recorded. The comparison between these two solutions (D_4 and D_5) is illustrated in Figure 5.28 and the overhead produced by D5 with respect to D4 is presented in Table 5.11.



(**Ho**: Number of hosts /**Np** : Number of process / ⊛: NFS storage node 0)

FIGURE 5.28: Comparison of design 4 (Greedy L5) and design 5 (Greedy L5 Blockchain) to process 1024 data.

Figure 5.28 illustrates that there is evidently a small overhead when the CV component is added to Design 4 (Greedy L5). However, each overhead (see Table 5.11) is calculated by applying the two solutions (D_4 and D_5) with exactly the same configuration (number of hosts and number of processes). This indicates, as can be seen in Table 5.11, that although the highest overhead is obtained in the configurations with 6 hosts regardless of the number of processes, a higher performance is obtained with these configurations (6 Hosts with Np 4 and 8) incorporating the CV component than the rest of the configurations of the solution that does not consider the verifiability component (D_4).

Configurations	**Design4**	**Design5**	**Overhead**	**% Overhead**
6 Hosts-Np4	1142.28	**1316.95**	0.1529	15.29%
3 Hosts-Np4	2164.36	2263.52	0.0458	4.58%
1 Host-Np4-node7	5111.25	5262.14	0.0295	2.95%
1 Host-Np4-node0	1904.58	2136.98	0.1220	12.20%
6 Hosts-Np8	1430.25	**1466.41**	0.0253	2.53%
3 Hosts-Np8	2523.54	2753.96	0.0913	9.13%
1 Host-Np8-node7	4057.23	4130.85	0.0181	1.81%
1 Host-Np8-node0	2157.93	2280.78	0.0569	5.69%

TABLE 5.11: Overhead of the D_5 (GM-CD/CV: Greedy L5 Blockchain) solution compared to D_4 (GM-CD/CV: Greedy L5).

As an example, the distributed configuration with the CV verifiability component with 6 hosts and 4 processes per stage in the workflow obtains better performance than the local solution of 1 host and the same number of processes (1 Host-Np4-node0) of the solution that does not have the verifiability component (D_4). In other words, the overhead of this distributed configuration (6-Hosts-Np4) with respect to the local configuration (1 Host-Np4-node0) applying the same number of processes (Np=4) is non-existent (-30.85%).

The comparison of the previous example shows that the application of the different parallelism and processing techniques in the CD/CV system allows to obtain higher performance with respect to solutions that do not consider the verification component.

5.5 Summary

In this chapter we have presented the evaluation of our proposed solutions by using three use cases, related to distributed systems, real-time systems data ingestion to blockchain, and analysis of CD/CV using and HPC workflow manager.

The results presented, not only verify the execution and sequence of the processing stages (CD model), but also the transactions (data exchange) performed by participants when processing assets (e.g., IoT data). Moreover, the results presented show the scalability and good performance of the solutions proposed in several environments.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

6.1 Introduction

The research work introduces a new system called GM-CD/CV, which acts as a middleware between workflow engines and blockchains. This system ensures the continuous verification and validation of the execution of IoT dataflow in a specific order, as well as the integrity of digital assets. It offers a reliable and transparent solution for real-time sensor data, with a focus on security, verification, and traceability.

6.2 Goals achievements

The thesis has fulfilled all the objectives proposed at the begging of the research.

O1 . To design a content delivery model for processing big IoT data in Edge-Fog-Cloud computing by using micro/nanoservice composition.

O2 . To design a continuous verification model based on blockchain to register significant events from the continuous delivery model, selecting techniques to integrate blockchain in quasi-real systems that allow ensuring traceability and non-repudiation of data obtained from devices and sensors.

O3 . To enhance the performance of blockchain systems to log IoT data to cope with the poor performance of current systems.

O4 . To evaluate the solution proposed with study cases to demonstrate its feasibility.

6.3 Contributions of the thesis

This research work introduces a novel model for IoT-edge-fog-cloud systems, which enhances the current state of the art by integrating Continuous Delivery/Continuous Verifiability and blockchain technology. The system is designed to enable automatic deployment of smart contracts at different stages of the workflow to trigger actions related to continuous verifiability. One of the key technical achievements is the real-time recording of transactions in blockchain. The major contributions of this thesis are the implementation of the model and its optimization to improve the data-flow processing and event registration in blockchain.

1. **Mathematical model:** We have created a mathematical model to illustrate the different parts and elements involved in workflow that utilize blockchain and real-time data. The model represents the various stages, such as definition, deployment, data acquisition, and contract verification, and the entities that participate in these stages.

2. **Distributed and automatic deployment:** The proposed solution in this research work enables the deployment and implementation of the model in an automated way, using tools such as Docker and Hyperledger Fabric. This integration allows for both local and distributed infrastructures to be used.

3. **Blockchain platform independent optimization techniques:** We suggest the use of optimization techniques for blockchain platforms that are independent of any specific platform. These techniques include atomic transactions and grouped validation, which are applied on top of the blockchain. By implementing these techniques, improvements can be made in the data collection transaction protocol and the data storage procedure, without being limited to a particular blockchain platform.

4. **Improved blockchain transactions validation performance:** We have made optimizations to improve the performance and scalability of blockchain transactions validation. These optimizations have resulted in faster validation speeds, and consistent data storage in near real-time. Consequently, the data is available for queries and continuous validation of the contract.

5. **Continuous verification of contracts:** The practice of periodically verifying contracts has been shown to be beneficial in large supply chain networks. This technique provides near real-time notification of contract breaches and can increase scalability.

In addition to the benefits mentioned earlier, the proposal presented in this research work also offers advantages for business logic actors. These advantages include.

- **Usability of the solution:** The proposed system in this research work includes a web interface that integrates the operational model and makes it accessible to companies or organizations without specialized knowledge in information technology. This interface guides users through the process of deploying their solution, setting up periodic contract verification, and performing verification queries. It is designed to be user-friendly for individuals without prior experience in blockchain or supply chains.

- **Faster conflicts resolution:** If a conflict arises between two or more participants in the network regarding a contract breach, the proposed system's web service can identify the recorded issues during the execution of the contracts in real-time. This helps establish an irrefutable party responsible for the problem, considering that the records are immutable. As a result, the system can help resolve disputes quickly and accurately.
- **Reliable verification with third parties:** The proposal in this research work includes the definition of client nodes that use containers to connect to the blockchain network. This design allows external entities, such as auditors, government agencies, or the police, to perform a reliable and traceable verification of all the registered activities of each organization on the blockchain network. This feature helps to ensure transparency and accountability in the network, which can be beneficial in various contexts.

In order to evaluate the proposed solution's performance and reliability, a series of experiments (uses cases) were conducted by the authors. These experiments were designed to measure the system's effectiveness in handling various tasks and scenarios, and to assess its ability to meet the demands of real-world workflows processes. The results of these experiments provide valuable insights into the strengths and limitations of the proposed solution, and help to inform potential improvements and future research directions.

The evaluation results indicate that the proposed optimizations allow for processing data requests at a higher registration rate than the default solution provided in blockchain platforms such as Hyperledger. Additionally, the proposed solution reduces the risk of missing data and minimizes transaction processing delays, which improves the overall reliability of the IoT dataflows processes.

We conducted an experimental evaluation of proposed solution applied to a mobility use case in the first scenario. The results of this evaluation indicate that the proposed solution is both feasible and highly performant, even when compared to a state-of-the-art workflow manager that does not include blockchain logging. This suggests that the proposed solution could be a viable alternative for organizations seeking to improve their workflows management processes, especially those in the mobility sector.

We believe that the GM-CD/CV model has great potential for organizations that are dealing with large amounts of data generated by IoT devices. With the exponential increase in data volume, this model can help establish continuous verifiability during IoT data processing workflows. However, while the model proposed is general in nature, its feasibility for multi-organization scenarios needs to be evaluated further by testing it with more datasets and use cases from different business sectors. This will help determine the extent to which the model can be effectively applied in various real-world scenarios.

6.4 Publications and conferences

1. Martinez-Rendon, Cristhian, et al. CD/CV: Blockchain-based schemes for continuous verifiability and traceability of IoT data for edge–fog–cloud. Information Processing & Management, 2023, vol. 60, no 1, p. 103155. Q1.

2. Martinez-Rendon, Cristhian, et al. On the continuous contract verification using blockchain and real-time data. Cluster Computing, 2022, p. 1-23. Q1.

3. Lopez-Arevalo, Ivan, et al. A wot-based method for creating digital sentinel twins of IOT devices. Sensors, 2021, vol. 21, no 16, p. 5531. Q1.

4. YANG, D., Martinez, C., Visuña, L. et al. Detection and analysis of COVID-19 in medical images using deep learning techniques. Sci Rep 11, 19638 (2021). https://doi.org/10.1038/s41598-021-99015-3. Q1.

5. Jesus Carretero and Cristhian Martinez: Blockchain-based schemes for continuous verifiability and traceability of IoT data. Workshop BDCSA2023: Big Data Convergence: from Sensors to Applications. 31st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP 2023). March 1-3, Naples, Italy.

6. Cristhian Martinez-Rendon, Dante D. Sánchez-Gallegos, Jesús Carretero, Jose L. Gonzalez-Compean , Antonio J. Rubio-Montero y Hernán Asorey. Calibración y evaluación experimental del Gestor CD/CV. Workshop CABAHLA. Jornadas Sarteco 2022. Alicante. 21 al 23 de septiembre de 2022.

7. Jesus Carretero, Cristhian Martinez, Jose L. Gonzalez & Dante Sanchez. CD/CV: Blockchain-based schemes for continuous verifiability and traceability of IoT data for edge-fog-cloud. Information Processing & Management Conference 2022. 20-21 October 2022. Wuhan. China. Online presentation.

6.5 Future research lines

Throughout the course of our research, we were able to identify a range of potential improvements that could be implemented within the proposed system in the future. These enhancements represent valuable opportunities for further development and refinement, with the potential to increase the efficiency, functionality, and overall effectiveness of the system. With careful consideration and strategic implementation, these future improvements could result in a more robust and optimized system that meets the evolving needs and expectations of different organizations.

1. **High-Performance:** In order to maintain each individual process and its corresponding stages within the file system of the proposed infrastructure, it is essential to implement a hash function method when constructing the transaction record. However, it is important to note that this process of preserving results incurs a certain cost in terms of reading and writing, which directly impacts the overall response time experienced by the user. To minimize these costs, we suggest exploring the option of exchanging information between stages from memory. Nevertheless, it is crucial to conduct a thorough study on how to ensure that continuous verifiability is maintained when adopting this approach.

2. **Geographically distributed multi-organization scenario:** The aim of this item is to define and assess a scenario in which multiple organizations are geographically dispersed. Since the proposed solution's components are contained within virtual containers, allowing for deployment across various infrastructures (such as edge, fog, and cloud), we could examine the costs (including performance, latency, and others) associated with distributing the solution in scenarios where the

nodes are completely dispersed geographically. Our primary interest lies in evaluating the performance of the GM-CD/CV system in this context, where each organization's stages and peer nodes are located in completely different geographic regions, forming a verifiable network. The aim is provide valuable insights into the feasibility and potential benefits of deploying GM-CD/CV in a geographically distributed multi-organization scenario.

3. **High availability scheme:** The implementation of a high availability scheme to ensure uninterrupted operation of GM-CD/CV. Without such a scheme, any failure in the CD service could potentially halt the continuous delivery process, while any failure in the CV services (peer nodes) could interrupt or even disable the verifiability network's records. Therefore, it is important to incorporate a robust high availability mechanism that can mitigate the risk of system failure and ensure the smooth functioning of the GM-CD/CV system at all times.

4. **Dynamic workflows:** Dynamic workflows that change over time (Varying loads and actors). The proposed solution considers a static workflow where the participating entities or actors are previously defined. In the same sense, a change of load or actors in the initial workflow may require the inclusion of more actors or components in the verifiability network to allow verification and validation on the new components.

5. **Improve maintenance of the supply chain elements:** Through the continuous verification of the blockchain's records, we intend to utilize the information to generate alerts for corrective maintenance and provide statistics or recommendations for predictive maintenance. This approach will enable us to identify components that are causing losses (such as sensors or trucks in poor condition) and establish

appropriate measures to address these issues. By utilizing the verifiability network and the data collected from IoT devices, we aim to optimize the maintenance processes for supply chain elements, which will ultimately improve the overall efficiency and effectiveness of the system.

BIBLIOGRAPHY

[1] Abubaker Haddud, Arthur DeSouza, Anshuman Khare, and Huei Lee. Examining potential benefits and challenges associated with the internet of things integration in supply chains. *Journal of Manufacturing Technology Management*, 2017.

[2] Bhabendu Kumar Mohanta, Soumyashree S Panda, and Debasish Jena. An overview of smart contract and use cases in blockchain technology. In *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–4. IEEE, 2018.

[3] D. Roy, D. Bhadra, and B. Das. Is blockchain the future of supply chain management?-a review paper. In *Proceedings of International Ethical Hacking Conference 2019*, pages 83–103. Springer Singapore, 2019.

[4] Peter Gonczol, Panagiota Katsikouli, Lasse Herskind, and Nicola Dragoni. Blockchain implementations and use cases for supply chains-a survey. *Ieee Access*, 8:11856–11871, 2020.

[5] Amitangshu Pal and Krishna Kant. Using blockchain for provenance and traceability in internet of things-integrated food logistics. *Computer*, 52(12):94–98, 2019.

[6] Moutaz Alazab, Salah Alhyari, Albara Awajan, and Ayman Bahjat Abdallah. Blockchain technology in supply chain management: an empirical study of the factors affecting user adoption/acceptance. *Cluster Computing - The Journal of Networks, Software Tools and Applications*, pages 1–19, November 2020. URL `https://doi.org/10.1007/s10586-020-03200-4`.

[7] Ewa Deelman, Karan Vahi, Mats Rynge, Rajiv Mayani, Rafael Ferreira da Silva, George Papadimitriou, and Miron Livny. The evolution of the pegasus workflow management software. *Computing in Science & Engineering*, 21(4):22–36, 2019.

[8] G Kousalya, P Balakrishnan, and C Pethuru Raj. Workflow management systems. In *Automated Workflow Scheduling in Self-Adaptive Clouds*, pages 55–64. Springer, 2017.

[9] JL Gonzalez-Compean, Victor Sosa-Sosa, Arturo Diaz-Perez, Jesus Carretero, and Jedidiah Yanez-Sierra. Sacbe: A building block approach for constructing efficient and flexible end-to-end cloud storage. *Journal of Systems and Software*, 135:143–156, 2018.

[10] Raffaele Montella, Diana Di Luccio, and Sokol Kosta. Dagon*: Executing direct acyclic graphs as parallel jobs on anything. In *2018 IEEE/ACM Workflows in Support of Large-Scale Science (WORKS)*, pages 64–73. IEEE, 2018.

[11] Yadu Babuji, Anna Woodard, Zhuozhao Li, Daniel S Katz, Ben Clifford, Rohan Kumar, Lukasz Lacinski, Ryan Chard, Justin M Wozniak, Ian Foster, et al. Parsl: Pervasive parallel programming in python. In *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing*, pages 25–36, 2019.

[12] Joseph Louis and Phillip S Dunston. Integrating iot into operational workflows for real-time and automated decision-making in repetitive construction operations. *Automation in Construction*, 94:317–327, 2018.

[13] Matteo Nardelli, Stefan Nastic, Schahram Dustdar, Massimo Villari, and Rajiv Ranjan. Osmotic flow: Osmotic computing+ iot workflow. *IEEE Cloud Computing*, 4(2):68–75, 2017.

[14] Raffaele Montella, Mario Ruggieri, and Sokol Kosta. A fast, secure, reliable, and resilient data transfer framework for pervasive iot applications. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 710–715. IEEE, 2018.

[15] Yinhao Li, Devki Nandan Jha, Gagangeet Singh Aujla, Graham Morgan, Albert Y Zomaya, and Rajiv Ranjan. Iotwc: Analytic hierarchy process based internet of things workflow composition system. In *2020 IEEE International Conference on Cloud Engineering (IC2E)*, pages 1–10. IEEE, 2020.

[16] Yanling Shao, Chunlin Li, and Hengliang Tang. A data replica placement strategy for iot workflows in collaborative edge and cloud environments. *Computer Networks*, 148:46–59, 2019.

[17] Farshad Firouzi, Bahar Farahani, and Alexander Marinšek. The convergence and interplay of edge, fog, and cloud in the ai-driven internet of things (iot). *Information Systems*, page 101840, 2021.

[18] M Adel Serhani, Hadeel T El-Kassabi, Khaled Shuaib, Alramzana N Navaz, Boualem Benatallah, and Amine Beheshti. Self-adapting cloud services orchestration for fulfilling intensive sensory data-driven iot workflows. *Future Generation Computer Systems*, 108:583–597, 2020.

[19] Awais Ahmad, Salvatore Cuomo, Wei Wu, and Gwanggil Jeon. Intelligent algorithms and standards for interoperability in internet of things, 2019.

[20] Rawaa Qasha, Jacek Cala, and Paul Watson. Dynamic deployment of scientific workflows in the cloud using container virtualization. In *2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 269–276. IEEE, 2016.

[21] Sina Shahhosseini, Arman Anzanpour, Iman Azimi, Sina Labbaf, DongJoo Seo, Sung-Soo Lim, Pasi Liljeberg, Nikil Dutt, and Amir M Rahmani. Exploring computation offloading in iot systems. *Information Systems*, page 101860, 2021.

[22] Esteban Municio, Johann Marquez-Barja, Steven Latré, and Stefano Vissicchio. Whisper: Programmable and flexible control on industrial iot networks. *Sensors*, 18(11):4048, 2018.

[23] Chris Simpkin, Ian Taylor, Daniel Harborne, Graham Bent, Alun Preece, and Raghu K Ganti. Efficient orchestration of node-red iot workflows using a vector symbolic architecture. *Future Generation Computer Systems*, 111:117–131, 2020.

[24] Xixun Yu, Zheng Yan, and Athanasios V Vasilakos. A survey of verifiable computation. *Mobile Networks and Applications*, 22(3):438–453, 2017.

[25] Konstantinos Christidis and Michael Devetsikiotis. Blockchains and smart contracts for the internet of things. *Ieee Access*, 4:2292–2303, 2016.

[26] Amir Taherkordi and Peter Herrmann. Pervasive smart contracts for blockchains in iot systems. In *Proceedings of the 2018 International Conference on Blockchain Technology and Application*, pages 6–11, 2018.

[27] Tiago M Fernández-Caramés and Paula Fraga-Lamas. A review on the use of blockchain for the internet of things. *Ieee Access*, 6:32979–33001, 2018.

[28] Xuanmei Qin, Yongfeng Huang, Zhen Yang, and Xing Li. Lbac: A lightweight blockchain-based access control scheme for the internet of things. *Information Sciences*, 554:222–235, 2021.

[29] Guangshun Li, Xinrong Ren, Junhua Wu, Wanting Ji, Haili Yu, Jiabin Cao, and Ruili Wang. Blockchain-based mobile edge computing system. *Information Sciences*, 561:70–80, 2021.

[30] Pushpa Singh and Narendra Singh. Blockchain with iot and ai: A review of agriculture and healthcare. *International Journal of Applied Evolutionary Computation (IJAEC)*, 11(4):13–27, 2020.

[31] Daniel Bumblauskas, Arti Mann, Brett Dugan, and Jacy Rittmer. A blockchain use case in food distribution: Do you know where your food has been? *International Journal of Information Management*, 52:102008, 2020.

[32] Daniele Mazzei, Giacomo Baldi, Gualtiero Fantoni, Gabriele Montelisciani, Antonio Pitasi, Laura Ricci, and Lorenzo Rizzello. A blockchain tokenizer for industrial iot trustless applications. *Future Generation Computer Systems*, 105:432–445, 2020.

[33] Ana Reyna, Cristian Martín, Jaime Chen, Enrique Soler, and Manuel Díaz. On blockchain and its integration with iot. challenges and opportunities. *Future generation computer systems*, 88:173–190, 2018.

[34] Alfonso Panarello, Nachiket Tapas, Giovanni Merlino, Francesco Longo, and Antonio Puliafito. Blockchain and iot integration: A systematic survey. *Sensors*, 18(8):2575, 2018.

[35] Quanyu Zhao, Siyi Chen, Zheli Liu, Thar Baker, and Yuan Zhang. Blockchain-based privacy-preserving remote data integrity checking scheme for iot information systems. *Information Processing & Management*, 57(6):102355, 2020. ISSN 0306-4573. doi: https://doi.org/10.1016/j.ipm.2020.102355. URL `https://www.sciencedirect.com/science/article/pii/S0306457320308505`.

[36] Bharat Bhushan, Chinmayee Sahoo, Preeti Sinha, and Aditya Khamparia. Unification of blockchain and internet of things (biot): requirements, working model, challenges and future directions. *Wireless Networks*, pages 1–36, 2020.

[37] Yeşem Kurt Peker, Xavier Rodriguez, James Ericsson, Suk Jin Lee, and Alfredo J Perez. A cost analysis of internet of things sensor data storage on blockchain via smart contracts. *Electronics*, 9(2):244, 2020.

[38] Arezou Naghib, Nima Jafari Navimipour, Mehdi Hosseinzadeh, and Arash Sharifi. A comprehensive and systematic literature review on the big data management techniques in the internet of things. *Wireless Networks*, pages 1–60, 2022.

[39] Aisha Siddiqa, Ibrahim Abaker Targio Hashem, Ibrar Yaqoob, Mohsen Marjani, Shahabuddin Shamshirband, Abdullah Gani, and Fariza Nasaruddin. A survey of big data management: Taxonomy and state-of-the-art. *Journal of Network and Computer Applications*, 71:151–166, 2016.

[40] Wu He, Feng-Kwei Wang, and Vasudeva Akula. Managing extracted knowledge from big social media data for business decision making. *Journal of Knowledge Management*, 2017.

[41] Pietro Pinoli, Stefano Ceri, Davide Martinenghi, and Luca Nanni. Metadata management for scientific databases. *Information Systems*, 81:1–20, 2019.

[42] Jay Lofstead, Joshua Baker, and Andrew Younge. Data pallets: containerizing storage for reproducibility and traceability. In *International Conference on High Performance Computing*, pages 36–45. Springer, 2019.

[43] Carole Goble, Sarah Cohen-Boulakia, Stian Soiland-Reyes, Daniel Garijo, Yolanda Gil, Michael R Crusoe, Kristian Peters, and Daniel Schober. Fair computational workflows. *Data Intelligence*, 2(1-2): 108–121, 2020.

[44] Bassirou Diène, Joel JPC Rodrigues, Ousmane Diallo, EL Hadji Malick Ndoye, and Valery V Korotaev. Data management techniques for internet of things. *Mechanical Systems and Signal Processing*, 138: 106564, 2020.

[45] Hui Li, Lishuang Pei, Dan Liao, Xiong Wang, Du Xu, and Jian Sun. Bddt: use blockchain to facilitate iot data transactions. *Cluster Computing - The Journal of Networks, Software Tools and Applications*, 23:1–21, May 2020. URL `https://doi.org/10.1007/s10586-020-03119-w`.

[46] Umair Khalid, Muhammad Asim, Thar Baker, Patrick CK Hung, Muhammad Adnan Tariq, and Laura Rafferty. A decentralized lightweight blockchain-based authentication mechanism for iot systems. *Cluster Computing - The Journal of Networks, Software Tools and Applications*, 23:2067–2087, September 2020. URL `https://doi.org/10.1007/s10586-020-03058-6`.

[47] Omar Alfandi, Salam Khanji, Liza Ahmad, and Asad Khattak. A survey on boosting iot security and privacy through blockchain. *Cluster Computing - The Journal of Networks, Software Tools and Applications*, 23:1–19, October 2020. URL `https://doi.org/10.1007/s10586-020-03137-8`.

[48] Vikram Puri, Ishaani Priyadarshini, Raghvendra Kumar, and Chung Van Le. Smart contract based policies for the internet of things. *Cluster Computing - The Journal of Networks, Software Tools and Applications*, 24:1–20, January 2021. URL `https://doi.org/10.1007/s10586-020-03216-w`.

[49] Manoharan Ramachandran, Niaz Chowdhury, Allan Third, John Domingue, Kevin Quick, and Michelle Bachler. Towards complete decentralised verification of data with confidentiality: Different ways to connect solid pods and blockchain. In *Companion Proceedings of the Web Conference 2020*, WWW '20, page 645–649, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450370240. doi: 10.1145/3366424.3385759. URL `https://doi.org/10.1145/3366424.3385759`.

[50] Petri Helo and AHM Shamsuzzoha. Real-time supply chain—a blockchain architecture for project deliveries. *Robotics and Computer-Integrated Manufacturing*, 63:101909, 2020.

[51] Christoph H.-J. Braun, Janina Traue, Boris Lingl, and Tobias Käfer. Documenting the execution of semantically modelled inter-organisational workflows on a distributed ledger. In *2021 IEEE International Conference on Blockchain (Blockchain)*, pages 280–286, 2021. doi: 10.1109/Blockchain53845.2021.00045.

[52] Manan Shukla, Jianjing Lin, and Oshani Seneviratne. Blockchain and iot enhanced clinical workflow. In *International Conference on Artificial Intelligence in Medicine*, pages 407–411. Springer, 2022.

[53] Mehrdokht Pournader, Yangyan Shi, Stefan Seuring, and SC Lenny Koh. Blockchain applications in supply chains, transport and logistics: a systematic review of the literature. *International Journal of Production Research*, 58(7):2063–2081, 2020.

[54] Iuon-Chang Lin and Tzu-Chun Liao. A survey of blockchain security issues and challenges. *IJ Network Security*, 19(5):653–659, 2017.

[55] Ethereum. Ethereum, 2020. URL `https://ethereum.org/`.

[56] Satoshi Nakamoto et al. Bitcoin: A peer-to-peer electronic cash system. *Working Paper*, 2008.

[57] Ashish Gadnis, Jeffrey A Keiser, Michael Linton, and Stanislav Natalenko. Blockchain-based identity and transaction platform, October 4 2018. URL `https://banqu.co/`. US Patent App. 15/767,969.

[58] Peter Verhoeven, Florian Sinn, and Tino T Herden. Examples from blockchain implementations in logistics and supply chain management: exploring the mindful use of a new technology. *Logistics*, 2(3): 20, 2018.

[59] Ethan Heilman, Foteini Baldimtsi, and Sharon Goldberg. Blindly signed contracts: Anonymous on-blockchain and off-blockchain bitcoin transactions. In *International conference on financial cryptography and data security*, pages 43–60. Springer, 2016.

[60] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert Van Renesse. Bitcoin-ng: A scalable blockchain protocol. In *13th {USENIX} symposium on networked systems design and implementation ({NSDI} 16)*, pages 45–59, 2016.

[61] Hyperledger-Fabric. Hyperledger fabric, 2020. URL `https://www.hyperledger.org/projects/fabric`.

[62] Dominique Guegan. Public blockchain versus private blockhain. Technical report, Centre d'Economie de la Sorbonne, 2017.

[63] Julia Nathan and Bob Jacobs. Blockchain consortium networks: Adding security and trust in financial services. *Journal of Corporate Accounting & Finance*, 31(2):29–33, 2020.

[64] Jiawen Kang, Rong Yu, Xumin Huang, Maoqiang Wu, Sabita Maharjan, Shengli Xie, and Yan Zhang. Blockchain for secure and efficient data

sharing in vehicular edge computing and networks. *IEEE Internet of Things Journal*, 6(3):4660–4670, 2018.

[65] Martin Valenta and Philipp Sandner. Comparison of ethereum, hyperledger fabric and corda. *no. June*, pages 1–8, 2017.

[66] P Sajana, M Sindhu, and M Sethumadhavan. On blockchain applications: hyperledger fabric and ethereum. *International Journal of Pure and Applied Mathematics*, 118(18):2965–2970, 2018.

[67] Sara Saberi, Mahtab Kouhizadeh, Joseph Sarkis, and Lejia Shen. Blockchain technology and its relationships to sustainable supply chain management. *International Journal of Production Research*, 57 (7):2117–2135, 2019.

[68] Xiaoqiong Xu, Gang Sun, Long Luo, Huilong Cao, Hongfang Yu, and Athanasios V. Vasilakos. Latency performance modeling and analysis for hyperledger fabric blockchain network. *Information Processing & Management*, 58(1):102436, 2021. ISSN 0306-4573. doi: https://doi.org/10.1016/j.ipm.2020.102436. URL `https://www.sciencedirect.com/science/article/pii/S0306457320309298`.

[69] Sunny Pahlajani, Avinash Kshirsagar, and Vinod Pachghare. Survey on private blockchain consensus algorithms. In *2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT)*, pages 1–6. IEEE, 2019.

[70] Rainer Böhme, Nicolas Christin, Benjamin Edelman, and Tyler Moore. Bitcoin: Economics, technology, and governance. *Journal of economic Perspectives*, 29(2):213–38, 2015.

[71] Sinclair Davidson, Primavera De Filippi, and Jason Potts. Economics of blockchain. *Available at SSRN 2744751*, 2016.

[72] Yaolin Zhang. Developing cross-border blockchain financial transactions under the belt and road initiative. *The Chinese Journal of Comparative Law*, 2020.

[73] J. Leung and J. Lee. 300cubits: Blockchain for shipping., 2017. URL `https://www.300cubits.tech/`.

[74] Rana M Amir Latif, Muhammad Farhan, Osama Rizwan, Majid Hussain, Sohail Jabbar, and Shahzad Khalid. Retail level blockchain transformation for product supply chain using truffle development platform. *Cluster Computing*, pages 1–16, 2020.

[75] Reshma Kamath. Food traceability on blockchain: Walmart's pork and mango pilots with ibm. *The Journal of the British Blockchain Association*, 1(1):3712, 2018.

[76] Ricardo Caballero and Braian Rivera. Blockchain: An alternative to enable traceability in the agricultural supply chain in panama. In *2019 7th International Engineering, Sciences and Technology Conference (IESTEC)*, pages 46–51. IEEE, 2019. doi: 10.1109/IESTEC46403.2019.00017.

[77] Samant Saurabh and Kushankur Dey. Blockchain technology adoption, architecture, and sustainable agri-food supply chains. *Journal of Cleaner Production*, 284:124731, 2021. ISSN 0959-6526. doi: https://doi.org/10.1016/j.jclepro.2020.124731. URL `http://www.sciencedirect.com/science/article/pii/S0959652620347752`.

[78] Stephen Wingreen, Ravishankar Sharma, et al. A blockchain traceability information system for trust improvement in agricultural supply chain. In *EUROPEAN CONFERENCE ON INFORMATION SYSTEMS (ECIS2019)*, 2019. URL `https://aisel.aisnet.org/ecis2019_rip/10/`.

[79] Thomas Bocek, Bruno B Rodrigues, Tim Strasser, and Burkhard Stiller. Blockchains everywhere-a use-case of blockchains in the pharma supply-chain. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 772–777. IEEE, 2017.

[80] Urvish Thakker, Ruhi Patel, Sudeep Tanwar, Neeraj Kumar, and Houbing Song. Blockchain for diamond industry: Opportunities and challenges. *IEEE Internet of Things Journal*, 2020.

[81] A. Quijano D. Jones, D. Kingston and C. Willette. Bext360., 2017. URL `https://www.bext360.com/`.

[82] Feng Tian. An agri-food supply chain traceability system for china based on rfid & blockchain technology. In *2016 13th international conference on service systems and service management (ICSSSM)*, pages 1–6. IEEE, 2016.

[83] Praveen Kumare Gopalakrishnan, John Hall, and Sara Behdad. Cost analysis and optimization of blockchain-based solid waste management traceability system. *Waste Management*, 120:594–607, 2021.

[84] Senthamiz Selvi Arumugam, Venkatesh Umashankar, Nanjangud C Narendra, Ramamurthy Badrinath, Anusha Pradeep Mujumdar, Jan Holler, and Aitor Hernandez. Iot enabled smart logistics using smart contracts. In *2018 8th International Conference on Logistics, Informatics and Service Sciences (LISS)*, pages 1–6. IEEE, 2018.

[85] Mohamed Ben-Daya, Elkafi Hassini, and Zied Bahroun. Internet of things and supply chain management: a literature review. *International Journal of Production Research*, 57(15-16):4719–4742, 2019.

[86] Tharaka de Vass, Himanshu Shee, and Shah J Miah. Iot in supply chain management: a narrative on retail sector sustainability. *International Journal of Logistics Research and Applications*, pages 1–20, 2020.

[87] Thanh Son Lam Nguyen, Guillaume Jourjon, Maria Potop-Butucaru, and Kim Loan Thai. Impact of network delays on hyperledger fabric. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 222–227. IEEE, 2019.

[88] Qassim Nasir, Ilham A Qasse, Manar Abu Talib, and Ali Bou Nassif. Performance analysis of hyperledger fabric platforms. *Security and Communication Networks*, 2018, 2018.

[89] Yuchen Wang, Shuang Li, Lei Xu, and Lizhen Xu. Improved raft consensus algorithm in high real-time and highly adversarial environment. In *International Conference on Web Information Systems and Applications*, pages 718–726. Springer, 2021.

[90] Wei Fu, Xuefeng Wei, and Shihua Tong. An improved blockchain consensus algorithm based on raft. *Arabian Journal for Science and Engineering*, 46(9):8137–8149, 2021.

[91] Kejiao Li, Hui Li, Hanxu Hou, Kedan Li, and Yongle Chen. Proof of vote: A high-performance consensus protocol based on vote mechanism & consortium blockchain. In *2017 IEEE 19th International Conference on High Performance Computing and Communications; IEEE 15th International Conference on Smart City; IEEE 3rd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 466–473. IEEE, 2017.

[92] Parth Thakkar, Senthil Nathan, and Balaji Viswanathan. Performance benchmarking and optimizing hyperledger fabric blockchain platform. In *2018 IEEE 26th international symposium on modeling, analysis, and simulation of computer and telecommunication systems (MASCOTS)*, pages 264–276. IEEE, 2018.

[93] Canhui Wang and Xiaowen Chu. Performance characterization and bottleneck analysis of hyperledger fabric. *arXiv preprint arXiv:2008.05946*, 2020.

[94] Haris Javaid, Chengchen Hu, and Gordon Brebner. Optimizing validation phase of hyperledger fabric. In *2019 IEEE 27th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 269–275. IEEE, 2019.

[95] Zhipeng Gao and Lulin Yang. Optimization scheme of consensus mechanism based on practical byzantine fault tolerance algorithm. In *CCF China Blockchain Conference*, pages 187–195. Springer, 2019.

[96] Chrysoula Stathakopoulou, Tudor David, and Marko Vukolić. Mir-bft: High-throughput bft for blockchains. *arXiv preprint arXiv:1906.05552*, 2019.

[97] Christian Gorenflo, Stephen Lee, Lukasz Golab, and Srinivasan Keshav. Fastfabric: Scaling hyperledger fabric to 20,000 transactions per second. In *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 455–463. IEEE, 2019.

[98] Takuya Nakaike, Qi Zhang, Yohei Ueda, Tatsushi Inagaki, and Moriyoshi Ohara. Hyperledger fabric performance characterization and optimization using goleveldb benchmark. In *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–9. IEEE, 2020.

[99] Saqib Ali, Guojun Wang, Bebo White, and Roger Leslie Cottrell. A blockchain-based decentralized data storage and access framework for pinger. In *2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE)*, pages 1303–1308. IEEE, 2018.

[100] Yacov Manevich, Artem Barger, and Yoav Tock. Endorsement in hyperledger fabric via service discovery. *IBM Journal of Research and Development*, 63(2/3):2–1, 2019.

[101] Ann Mary Joy. Performance comparison between linux containers and virtual machines. In *2015 International Conference on Advances in Computer Engineering and Applications*, pages 342–346. IEEE, 2015.

[102] Qi Zhang, Ling Liu, Calton Pu, Qiwei Dou, Liren Wu, and Wei Zhou. A comparative study of containers and virtual machines in big data environment. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pages 178–185. IEEE, 2018.

[103] D. Treat, G. Giordano, L. L. Schiatti, and H. Borne-Pons. Connecting ecosystems: Blockchain integration., 2018. URL `https://www.accenture.com/us-en/insights/blockchain/integration-ecosystems`.

[104] Valentina Armenise. Continuous delivery with jenkins: Jenkins solutions to implement continuous delivery. In *2015 IEEE/ACM 3rd International Workshop on Release Engineering*, pages 24–27. IEEE, 2015.

[105] Ewa Deelman, Gurmeet Singh, Mei-Hui Su, James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Karan Vahi, G Bruce Berriman, John Good, et al. Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming*, 13(3): 219–237, 2005.

[106] Andy B Yoo, Morris A Jette, and Mark Grondona. Slurm: Simple linux utility for resource management. In *Workshop on Job Scheduling Strategies for Parallel Processing*, pages 44–60. Springer, 2003.

[107] Dante Domizzi Sanchez-Gallegos, JL Gonzalez-Compean, Jesus Carretero, Heidy Marin, Andrei Tchernykh, and Raffaele Montella. Puzzlemesh: A puzzle model to build mesh of agnostic services for edge-fog-cloud. *IEEE Transactions on Services Computing*, 2022.

[108] Cristhian Martinez-Rendon, Diego Camarmas-Alonso, Jesus Carretero, and Jose L Gonzalez-Compean. On the continuous contract verification using blockchain and real-time data. *Cluster Computing*, pages 1–23, 2021.

[109] Arati Baliga, Nitesh Solanki, Shubham Verekar, Amol Pednekar, Pandurang Kamat, and Siddhartha Chatterjee. Performance characterization of hyperledger fabric. In *2018 Crypto Valley conference on blockchain technology (CVCBT)*, pages 65–74. IEEE, 2018.

[110] Suporn Pongnumkul, Chaiyaphum Siripanpornchana, and Suttipong Thajchayapong. Performance analysis of private blockchain platforms in varying workloads. In *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–6. IEEE, 2017.

[111] Sara Rouhani and Ralph Deters. Performance analysis of ethereum transactions in private blockchain. In *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pages 70–74. IEEE, 2017.

[112] J. L. Gonzalez, J. C. Perez, Vor J Sosa-Sosa, Luis M Sanchez, and B. Bergua. Skycds: A resilient content delivery service based on diversified cloud storage. *Simulation Modelling Practice and Theory*, 54: 64–85, 2015.

[113] Panos Vassiliadis. A survey of extract–transform–load technology. *International Journal of Data Warehousing and Mining (IJDWM)*, 5 (3):1–27, 2009.

[114] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference*, page 30. ACM, 2018.

[115] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151:1–32, 2014.

[116] Jorge Luis Ortega-Arjona. *Patterns for Parallel Software Design*. Wiley Publishing, 1st edition, 2010. ISBN 0470697342, 9780470697344.

[117] Deepa Pavithran, Khaled Shaalan, Jamal N Al-Karaki, and Amjad Gawanmeh. Towards building a blockchain framework for iot. *Cluster Computing*, 23:2067–2087, September 2020. URL `https://doi.org/10.1007/s10586-020-03059-5`.

[118] Peilin Zheng, Zibin Zheng, Xiapu Luo, Xiangping Chen, and Xuanzhe Liu. A detailed and real-time performance monitoring framework for blockchain systems. In *2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)*, pages 134–143. IEEE, 2018.

[119] Ali Dorri, Salil S Kanhere, and Raja Jurdak. Towards an optimized blockchain for iot. In *2017 IEEE/ACM Second International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pages 173–178. IEEE, 2017.

[120] Mike Folk, Gerd Heber, Quincey Koziol, Elena Pourmal, and Dana Robinson. An overview of the hdf5 technology suite and its applications. In *Proceedings of the EDBT/ICDT 2011 workshop on array databases*, pages 36–47, 2011.

[121] Dmitriy Morozov and Tom Peterka. Block-parallel data analysis with diy2. In *2016 IEEE 6th Symposium on Large Data Analysis and Visualization (LDAV)*, pages 29–36. IEEE, 2016.

[122] William Gropp, Ewing Lusk, Nathan Doss, and Anthony Skjellum. A high-performance, portable implementation of the mpi message passing interface standard. *Parallel computing*, 22(6):789–828, 1996.

[123] Raul Montoliu, Jan Blom, and Daniel Gatica-Perez. Discovering places of interest in everyday life from smartphone data. *Multimedia tools and applications*, 62(1):179–207, 2013.

[124] Yu Zheng, Hao Fu, Xing Xie, Wei-Ying Ma, and Quannan Li. Geolife gps trajectory dataset-user guide, july 2011. *URL: https://www. microsoft. com/en-us/research/publication/geolife-gps-trajectory-dataset-user-guide*, 2011.

[125] Wesley Mathew, Ruben Raposo, and Bruno Martins. Predicting future locations with hidden markov models. In *Proceedings of the 2012 ACM conference on ubiquitous computing*, pages 911–918, 2012.

[126] Hristijan Gjoreski, Mathias Ciliberto, Francisco Javier Ordoñez Morales, Daniel Roggen, Sami Mekki, and Stefan Valentin. A versatile annotated dataset for multimodal locomotion analytics with mobile devices. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, pages 1–2, 2017.

[127] Zhibin Xiao, Yang Wang, Kun Fu, and Fan Wu. Identifying different transportation modes from trajectory data using tree-based ensemble classifiers. *ISPRS International Journal of Geo-Information*, 6(2):57, 2017.

[128] Sina Shaham, Ming Ding, Bo Liu, Zihuai Lin, and Jun Li. Machine learning aided anonymization of spatiotemporal trajectory datasets.

In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–6. IEEE, 2019.

[129] Apache Kafka. Apache kafka. *A distributed streaming platform (accessed May 2019). https://kafka. apache. org/intro*, 2021.

[130] Hui Liu, Dan Chen, Da Chen, Xiyu Zhang, Huijie Li, Lipan Bian, Minglei Shu, and Yinglong Wang. A large-scale multi-label 12-lead electrocardiogram database with standardized diagnostic statements. *Scientific Data*, 9(1):272, 2022.