الجمهورية الجزائرية الديمقراطية الشعبية

وزارة التعليم العالي والبحث العلمي

جامعة سعيدة د. مولاي الطاهر

كلية العلوم

قسم: الإعلام الآلي

Master Thesis

Speciality: Computer Networks and Distributed Systems

Theme

Consistency Management Strategy for Data Replication in Fog Computing

Presented by :

Hanane TAIBI

Selamet BENTADJ

Supervisor :

Dr. Said LIMAM

Promotion 2023 - 2024

Abstract

Cloud computing provides scalability and cost-efficiency but poses risks like internet dependency and security vulnerabilities. Fog computing decentralizes resources to the network edge, enhancing latency, bandwidth efficiency, and real-time data processing, ideal for low-latency applications compared to centralized cloud models. Ensuring data consistency in this environment, particularly when accessing files with multiple replicas, requires access to the most recent version. We propose a strategy that improves this process by focusing on user access to specific parts of a file rather than the entire replica. By dividing files into blocks, each containing a specific version, users can access the latest version of only the required block, enhancing data consistency, performance, and availability in a distributed file system within the Fog Computing environment. To achieve this, we implemented rigorous synchronization methods. We explored both centralized and distributed approaches for managing data synchronization, detailing a process that allows efficient access to the latest available file version. This strategy ensures that users consistently access the most recent data, optimizing system performance and enhancing the overall user experience.

Résumé

Le Cloud Computing offre évolutivité et rentabilité, mais présente des risques comme la dépendance à l'internet et des vulnérabilités de sécurité. Le Fog Computing décentralise les ressources vers la périphérie du réseau, améliorant la latence, l'efficacité de la bande passante et le traitement des données en temps réel, idéal pour les applications à faible latence par rapport aux modèles cloud centralisés. Assurer la cohérence des données dans cet environnement, en particulier lors de l'accès à des fichiers avec plusieurs répliques, nécessite d'accéder à la version la plus récente. Nous proposons une stratégie qui améliore ce processus en se concentrant sur l'accès des utilisateurs à des parties spécifiques d'un fichier plutôt qu'à l'intégralité de la réplique. En divisant les fichiers en blocs, chacun contenant une version spécifique, les utilisateurs peuvent accéder à la dernière version du seul bloc requis, améliorant ainsi la cohérence des données, la performance et la disponibilité dans un système de fichiers distribué au sein de l'environnement Fog Computing. Pour y parvenir, nous avons mis en œuvre des méthodes de synchronisation

rigoureuses. Nous avons exploré des approches centralisées et distribuées pour gérer la synchronisation des données, détaillant un processus permettant un accès efficace à la dernière version disponible des fichiers. Cette stratégie garantit que les utilisateurs accèdent toujours aux données les plus récentes, optimisant les performances du système et améliorant l'expérience utilisateur globale.

الملخص

ملخص يوفر الحوسبة السحابية القابلية للتوسع والكفاءة من حيث التكلفة ولكنه يعرض المخاطر مثل الاعتماد على الإنترنت والثغرات الأمنية. تعمل الحوسبة الطرفية على لامركزية الموارد إلى حافة الشبكة، مما يعزز الكفاءة في النطاق الترددي وزمن الوصول ومعالجة البيانات في الوقت الفعلي، مما يجعلها مثالية للتطبيقات التي تتطلب زمن وصول منخفض مقارنة بالنماذج السحابية المركزية. إن ضمان اتساق البيانات في هذا البيئة، خاصة عند الوصول إلى الملفات التي تحتوي على نسخ متعددة، يتطلب الوصول إلى أحدث إصدار. نقترح استراتيجية تحسن هذه العملية من خلال التركيز على وصول المستخدمين إلى أجزاء معينة من الملف بدلاً من النسخة الكاملة. عن طريق تقسيم الملفات إلى كتل، كل منها يحتوي على إصدار محدد، يمكن للمستخدمين الوصول إلى أحدث إصدار من الكتلة المطلوبة فقط، مما يعزز اتساق البيانات والأداء والتوافر في نظام ملفات موزع داخل بيئة الحوسبة الطرفية. لتحقيق ذلك، قمنا بتنفيذ طرق مزامنة صارمة. اكتشفنا نهجاً مركزياً وموزعاً لإدارة مزامنة البيانات، مع تفصيل عملية تسمح بالوصول الفعال إلى أحدث إصدار متاح من الملفات. تضمن هذه الاستراتيجية وصول المستخدمين باستمرار إلى أحدث البيانات، مما يعزز أداء النظام ويحسن تجربة المستخدم الشاملة.

ACKNOWLEDGEMENTS

First of all, I thank Allah Almighty for giving me the strength and patience to finish my search. This study is dedicated from my heart to my beloved parents, who have inspired me and gave me strength when I thought about giving up, they provide me with spiritual, emotional and financial support. To my brothers, sisters, relatives, professors, friends and colleagues who shared with me their words of advice and encouragement to finish this study. I would like to express my thanks of gratitude to my Supervisor Dr LIMAM Said, for all the instructions, assistance, and supports to my graduate project. Especially, I deeply appreciate his patience with me. Furthermore, I would like to express my sincere thanks to myself for my great patience and effort.

Dedicate

I dedicate deeply this work to :

my grandfather "TAIBI Yahia", whose prayers still envelop me with protection and warmth, even in his absence.

my dear grandmother, a beacon of grace, wisdom, and unwavering and unconditional love. my beloved parents ,their unwavering love, sacrifices, and endless support have shaped me into the person I am today.

my sisters"Sara, Cheimaa, Aya, and Hanane", My journey wouldn't be as colorful and meaningful without them. In each moment, laughter, or tear shed

and finally to my slice of joy my nephew Yahia.

thanks from Selamet

Dedicate

Extraordinary things only happen to extraordinary people. This statement is not about exclusivity but rather a call to action – a challenge to elevate our thinking and efforts. It speaks to the dreamers who act on their visions, the innovators who break new ground, and the brave souls who push the boundaries of what's possible. A prime example is my grandfather . After turning fifty, he chose to memorize and teach the Qur'an, despite being illiterate. His story has always been a source of inspiration for me. May God grant him a place in heaven. Similarly, my grandmother endured so much to raise a generation of peaple. I am deeply thankful to God for her, as well as my parents who have shaped me into who I am today and my sisters and all my family Thank you all so much for evrything.

The people of Gaza and Palestine exemplify extraordinary resilience and strength. Their unwavering faith, fearing only God, has profoundly impacted us. They have demonstrated the pride that we lost when we strayed from Islam, while they remained steadfast in their adherence to its teachings. In summary, "While they win the test of patience, let us ensure we do not fail the test of gratitude".

Thanks from Hanane

ACRONYMS

ACRONYMS

IAAS : *Infrastructure As A Service*

PAAS : *Platform As A Service*

SaaS : *Software As A Service*

QoS : *Quality of Service*

IOT : *Internet Of Things*

ROWA : *Read One Write All*

CONTENTS

List of Tables

Table **Page**

List of Figures

CHAPTER 1

GENERAL INTRODUCTION

1.1 Introduction

1.1.1 Context

Cloud computing has revolutionized data and application management by providing scalable resources and services via the internet, offering benefits like cost efficiency and scalability. However, it suffers from drawbacks such as latency and bandwidth issues due to centralized data centers, which can hinder real-time applications and increase vulnerability to cyber-attacks.

In contrast, fog computing extends cloud capabilities to the network edge, closer to users. This proximity enables faster data processing, reduces latency, and supports real-time applications like IoT and autonomous vehicles. By processing data locally, fog computing minimizes bandwidth demands on central servers and enhances security and reliability through its decentralized architecture. A critical aspect of fog computing is data replication, which involves strategically duplicating data across distributed nodes to enhance system efficiency and reliability. This decentralized approach reduces latency, improves fault tolerance, and optimizes performance. Dynamic replication strategies adapt to changing conditions, and consistency models ensure data synchronization. Security measures are essential to protect replicated information, making data replication vital for achieving optimal performance and responsiveness at the network edge. In this thesis we will present our strategy for ensuring data consistency in a Fog Computing environment. Our objective is to ensure that each user always accesses the most recent version of the files. To achieve this, we ensure that users access the latest files through rigorous synchronization methods. Our approach involves dividing the replica into blocks, with each block containing a specific version.We propose two solutions: centralized and distributed. In the cen-

tralized approach each fog node will begin by calculating its local ideal vector based on the data replicas it holds, ensuring local consistency without constant communication with other nodes. This local ideal vector will then be sent to the cloud. The cloud, upon receiving the local ideal vectors from all fog nodes, will calculate a global ideal vector, providing a consolidated view of the system's state and facilitating the detection of inconsistencies and anomalies. The global ideal vector will then be sent back to all fog nodes to start the updating process. The synchronization can be triggered in three ways: a method based on the number of modifications, a periodic method, or a hybrid synchronization method. In a distributed synchronization approach, fog nodes will communicate directly with each other to avoid latency and bandwidth issues associated with centralized synchronization. Each fog node will construct a matrix reflecting the local data state, calculate a local ideal vector, and exchange these vectors with other nodes. Upon receiving the local ideal vectors from other nodes, each fog node will construct a global ideal vector, ensuring a coherent view of the data across all nodes. Finally, after verifying the completeness of the received vectors, each node will update its data to align with the global ideal vector, ensuring system-wide consistency.

To evaluate our strategy, we extended the IFogSim simulator to facilitate comparative analyses of distributed and centralized strategies. We conducted multiple series of experiments to gather data, analyze results, and interpret findings. Key metrics including execution time, energy consumption, and network usage are evaluated under varying conditions. This thesis is structured into four main chapters, summarized as follows:

1.2 Organization work

Our thesis consists of four chapters:

Chapter 01: In this chapter, we discussed cloud computing and its disadvantages, as well as fog computing and its advantages.

Chapter 02: Focus on data replication in fog computing, detailing how it strategically duplicates data across distributed nodes to enhance system efficiency, reliability, and consistency in distributed computing environments.

Chapter 03: This chapter will provide a detailed description of our proposed strategy.

Chapter 04: This final chapter will outline the implementation steps of the proposed approach. We will detail the evaluation study of this strategy. Experimental results will be interpreted. Finally, a summary and perspectives will conclude our work.

1

CHAPTER 2

CLOUD AND FOG COMPUTING

2.1 Introduction

Fog computing infrastructures play a crucial role in the Internet ecosystem. Leveraging their robust storage and computational capabilities, coupled with their proximity to end-users, they enable the processing of data close to its point of origin. This chapter serves to provide comprehensive definitions and explanations of key concepts utilized throughout this thesis. Additionally, it offers a detailed exploration of the evolution necessary to establish the fog computing infrastructure as we recognize it today.

2.2 Cloud computing

Cloud computing is an evolution of cluster and grid computing, which originally centralized resources for high-performance computing. The other advantage of cloud computing is its layered architecture that allows customers to purchase services at different levels of abstractions commonly known as IaaS, PaaS and SaaS [16]. Software as a Service (SaaS) delivers customer-facing applications like accounting software, database applications, and email services, with users interacting with the software without managing the underlying technical infrastructure of the cloud. Platform as a Service (PaaS) furnishes a development platform for application developers, offering programming environments and tools to create applications. Users have the ability to develop applications using these tools and configure and manage the technical aspects of the cloud platform. Infrastructure as a Service (IaaS) provides cloud infrastructure services such as managing servers, storage, and network devices. Users of IaaS have the flexibility to

manage, modify, or configure the cloud infrastructure according to their specific requirements. Though cloud computing has so many advantages, it also suffers from certain shortcomings too. These shortcomings include the requirement of high capacity (bandwidth) client access link, high latency and security. [17][9][12]

2.3 Disadvantages of Cloud Computing

2.3.1 The Bandwidth:

The bandwidth required to move this to the Cloud is enormous, and the costs would be so significant that it is more advantageous to purchase the storage ourselves rather than pay someone else to handle it.

2.3.2 Security:

Storing data in the cloud can pose security issues, as the data is stored on third-party servers that may be vulnerable to hacker attacks. Companies must take additional security measures to protect their data.

2.3.3 Availability:

Cloud computing relies on Internet availability. If your Internet connection is down, you cannot access your data stored in the cloud.

2.3.4 Costs:

Although cloud computing can be cost-effective, it depends on how you use it. If you use a lot of data, costs can quickly increase. Dependency on the Provider: When using cloud computing, you rely on the cloud service provider to store and manage your data. If the service provider experiences downtime or ceases its operations, you may lose access to your data. Confidentiality: Data stored in the cloud is often accessible to other users of the same cloud service. Companies must take measures to protect their sensitive and confidential data.

2.4 Fog computing

fog computing is a decentralized computing infrastructure that extends the capabilities of cloud computing by bringing computation, storage, and networking resources closer to the edge of the network, typically within the proximity of end-users and IoT devices. This architecture enables faster data processing, reduced latency, and improved efficiency by distributing computing tasks across a network of edge devices and fog nodes. Fog computing is more advanced and its performance better than cloud computing for handling user requests and emerging standards [20].

2

The term "Fog Computing" was introduced by the Cisco Systems as new model to ease wireless data transfer to distributed devices in the Internet of Things (IoT) network paradigm [8][4]. Thus fog computing can provide better quality of service in terms of delay, power consumption, reduced data traffic over the Internet etc [5]. Other similar concepts where the computing resources have been proposed to be located closer to the users to overcome the limitations of cloud computing include cloudlets and edge computing [14][1].

2.5 Characteristics of Fog Computing

- Low inertness, edge area and area mindfulness: Fog processing arrangement better administrations to end clients at the edge of the network.
- Geographical Distribution: Fog computing application, objective, and services are widely distributed.
- Real time interactions: speedy services need real time interaction in Fog computing.
- Heterogeneity: Fog computing supports heterogeneous devices and support nodes in a wide variety of environments.
- Interoperability: Fog gives a wide range of services so for that purpose Fog devices incorporate for streaming of services [10].
- Scalability: The cloud might become the bottleneck if the data generated by end devices are constantly transferred to it [3].

2.5.1 The functioning of fog computing

Fog nodes are installed near IoT (Internet of Things) devices or users that generate data. These nodes can be routers, switches, gateways, or IoT devices. Data is gathered from sensors, IoT devices, or nearby users, and then processed on local Fog nodes. Fog nodes can also store data locally to reduce latency and improve availability. Processed data can be transferred to the cloud for long-term storage, analysis or further processing. Fog computing applications are designed to run on local Fog nodes, rather than on remote servers in the cloud. This allows for a faster and more efficient response to data processing requests. Fog nodes can be managed and monitored remotely from a Fog management center, which can be located in the cloud or on-premises.

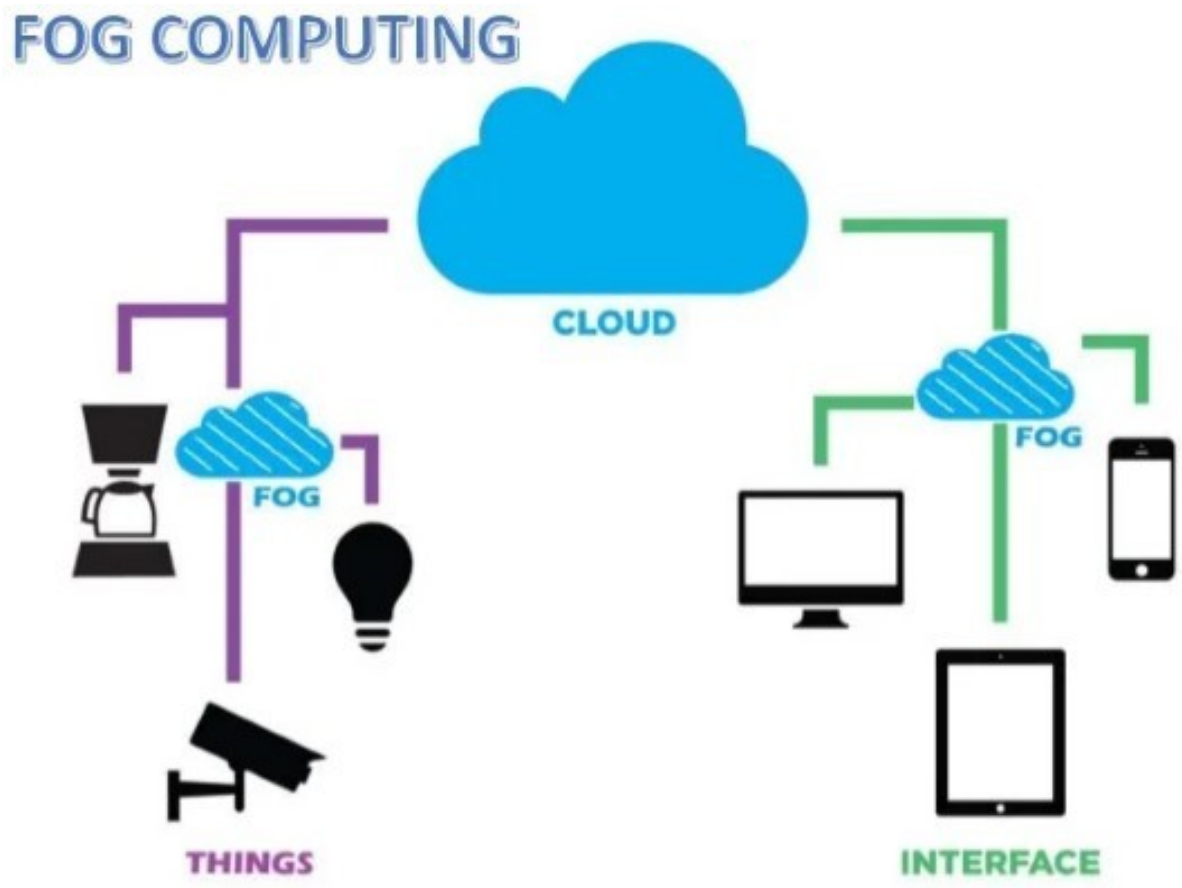

Figure 2.1: fog computing.

2.6 Cloud computing vs fog computing

Cloud computing is infrastructure based and required hardware and software to manage task and processing where Fog utilizes resources of devices on edge but it will not replace cloud computing at present time, which is top of all business and provide employment to the world. It was concluded that cloud computing and Fog computing have own characteristics to process data burdens and preferences but are the supplement to each other [18] In this section Fog and cloud computing comparison is given in table 2.1:

[11]

Parameters	Fog Computing	Cloud Computing
Goal	Enhance proficiency and execution of process that should be transported to the cloud for handling, investigation and storage	Give a request of greatness change in the practical, powerful provisioning of IT administrations
Abstraction Level	High	High
scalability Degree	High	High
Support of Multitask	Yes	Yes
Transparency Level	High	High
Run time	Real time services	Real time services
Transmission	Device to device	Device to Cloud
Infrastructure	Flexible	3 models (PaaS, IaaS, SaaS)
Resource	Unlimited	Unlimited
Type of service	CPU, network, memory, bandwidth, device, storage	IaaS, PaaS, SaaS, Everything as a service
Number of users	Unlimited	Unlimited
Security	Possible, determined	Undefined
Response Time Low High	Low	High

Table 2.1: Fog vs Cloud Computing.

3

2.7 Conclusion

Fog computing represents a major evolution in distributed computing, bringing data processing closer to the network edge to meet the requirements of modern applications. Its benefits include faster processing, reduced latency, and better resilience to network outages. However, it poses challenges in terms of managing distributed resources, data consistency, and security. Despite this, its potential to enhance performance and efficiency of applications makes it a promising technology for the future of distributed computing.

CHAPTER 3

DATA REPLICATION AND CONSISTENCY

3.1 Introduction

In fog computing , data replication involves strategically duplicating data across distributed nodes to enhance system efficiency and reliability. This decentralized approach reduces latency, improves fault tolerance, and optimizes performance. Dynamic replication strategies adapt to changing conditions, while consistency models govern data synchronization. Security measures are crucial for safeguarding replicated information. Overall, data replication in Fog Computing plays a pivotal role in achieving optimal performance and responsiveness at the network edge.

3.2 Definition

Data replication (duplication of data across multiple nodes) is an effective solution for achieving good performance and better data availability. The main objective of replication is to facilitate access to data, while increasing availability, either because the data is copied to different sites allowing requests to be distributed, or because a site can take over when the main server collapses. Parallel processing of access requests improves performance by letting users access data in a neighboring node to avoid accessing data in a remote site. Replication also provides better failure resilience and response time [13].

3.3 Types of replication

3.3.1 Synchronous replication

In fog computing, synchronous replication ensures that when a transaction updates a primary replica, all of its secondary replicas are also updated in the same transaction. This ensures that all data is kept at the same update level, ensuring that the latest version of the data is provided regardless of which replica is accessed. Disadvantages of synchronous replication in the context of fog computing include the need to manage resource-intensive transactions and the complexity of a site's failure management algorithms. This is why asynchronous replication is often preferred in fog computing.

3.3.2 Asynchronous replication

In fog computing, asynchronous replication implies that certain local sub-operations carried out following a global update are accomplished in independent transactions and deferred in time. Copy update transactions can be initiated as soon as possible or at predetermined times, such as evenings or weekends. The advantage of asynchronous replication in fog computing is the ability to update data at selected times, while allowing access to previous versions before upgrading. However, access to the latest version is not guaranteed, which can be a disadvantage in certain situations.

3.3.3 State-Based Approach

In this approach, the source replica is updated first. Then, the subsystem transmits the state of the updated replica to all other replicas by merging the delivered state with the local state of each replica. This ensures that all replicas are aligned with the most recent version of the data.

3.3.4 Operation-Based Approach

In this approach, the subsystem sends the update operation and its parameters to all replicas. Each replica then executes the operation locally to update its own data based on the provided parameters. This approach is often used when update operations are complex or when data needs to be updated transactionally across all replicas.

3.4 Advantages of replication

An improvement in reliability or operational safety:

- If one copy fails, it is still possible to obtain the data from another copy.

4

- Redundancy provides better protection against file corruption.

An enhancement in performance:

- Distributing the workload among multiple servers.
- Geographic scalability by bringing servers closer to clients.

Data availability:

- Data is available locally rather than through network connections; it is accessible locally even in the absence of a connection to a central server, ensuring users are not cut off from their data in case of long-distance network connection failure.

Response time:

Replication improves polling request response times for two reasons:Requests are processed on a local server without access to a wide area network, which accelerates throughput. Furthermore, local processing reduces the load on the central database server, which makes it possible to use less processor.

3.5 Replication issues

Despite all the advantages it provides, the replication technique raises a certain number of problems which we will address briefly in what follows [6].

Choice of data to replicate:
Determine which data should be replicated on fog nodes to ensure adequate availability and performance. Degree of replication: Define how many replicas of each data should be created to ensure sufficient availability and resilience of the system. **Replication timing:**
Choose the optimal time to replicate data based on the system workload and availability requirements.

Replica placement:
Determine where to place replicas in the fog to ensure optimal response times and efficient resource utilization.

Management of replica consistency:
Ensure data consistency among different replicas distributed across the fog, despite updates and distributed read/write operations. System partitioning: Manage network failures or partitions to ensure that data replicas converge to a consistent state and maintain system availability.

Choice of the best replica:
Select the best replica from a data consistency perspective during query processing to ensure accurate and reliable results.

3.6 Data Consistency Challenges and Approaches in Distributed Computing Environments: Grids, Cloud, and Fog Computing

3.6.1 Grid Computing

In recent years, there has been a significant increase in the volume of information processed, necessitating the development of large-scale networks and the geographical distribution of information worldwide. Consequently, researchers have conceptualized grids that leverage the computing power and storage capacity of multiple processing units interconnected by networks. In a grid, databases are numerous, autonomous (under strict local control), and highly heterogeneous in terms of size and complexity. Additionally, client stations can be mobile terminals that work in offline mode and occasionally synchronize with the databases on the network. Data management in such a context presents significant challenges, as techniques must scale while supporting new needs related to data autonomy, heterogeneity, and node mobility within the grid. Maintaining data consistency is one of the primary tasks in designing and managing databases. In a centralized system, this task is relatively easy because the data is located in a specific place, and updates are performed locally without competition, simplifying data management. However, in large-scale distributed systems like grids, managing consistency becomes challenging because data is distributed across geographically distant sites. The databases are fragmented across these sites, with some being replicated in multiple locations. These sites are interconnected, forming a network, and communicate via messages. Regardless of the chosen architecture, updates are performed through remote queries, and read/write requests are concurrent. Thus, update propagation time, fault tolerance, and concurrency management must be considered. Effective methods are needed to maintain data consistency during updates. In a grid, replication improves data availability and access performance. Generally, replicated data must meet two important quality criteria: freshness and consistency with the base data. Consistency is ensured when all copies of a data item are identical at any given moment. However, ensuring replica consistency in grids is a significant challenge. Numerous studies have been conducted in this area, revealing that maintaining strong replica consistency is costly in terms of system performance. To preserve performance, consistency must be relaxed, making the trade-off between consistency and performance a persistent challenge to overcome.

5

3.6.2 Cloud

Data storage in the Cloud requires a reliable and appropriate infrastructure to ensure that all resources can be used and shared efficiently, thereby reducing issues related to data consistency. Consistency management is defined as the control of access to provide a view that abstracts replication, distribution, and concurrent access to shared data. Effective management of data copies in terms of update propagation is necessary. The overhead induced by maintaining this consistency can significantly impact system performance. Therefore, it is crucial to ensure the mutual consistency of a set of replicas within acceptable timeframes. There are two types of consistency:

- Strong Consistency: Replicas exhibit strong consistency when any query to any copy reflects the result of all previous modifications.
- Weak Consistency: Replicas exhibit weak consistency if it is tolerated that a query may not reflect all previous modifications, with the guarantee that all changes will eventually be propagated within a finite period.

3.6.3 Fog Computing

Data consistency in the context of Fog Computing is a critical aspect to consider in distributed environments. Fog Computing extends the capabilities of Cloud Computing by deploying computational, storage, and processing resources closer to the network edges, thereby reducing latency and improving data usage efficiency. However, replicating data across multiple nodes in a Fog Computing environment introduces specific challenges regarding consistency. Here are some key points regarding data consistency in Fog Computing:

- **Low Latency:** Fog Computing can support real-time services (e.g., gaming, video streaming) by ensuring low latency.
- **Geographical Distribution and Large Scale:** Fog Computing can provide distributed computing and storage resources for large-scale and widely distributed applications [2]. Data consistency must be managed considering this distribution to ensure a unified and coherent view.
- **Flexibility and Heterogeneity:** Fog Computing enables collaboration across different physical environments and infrastructures among multiple services [19]. Consistency strategies must be adapted to account for this heterogeneity.
- **Bandwidth Constraints:** Some Fog nodes may have bandwidth constraints. Consistency mechanisms must minimize bandwidth usage while ensuring data consistency.

Data consistency in Fog Computing requires a thoughtful approach, taking into account the specific characteristics of this distributed environment. Designers must choose appropriate consistency models, replication mechanisms, and conflict management strategies to meet the requirements of applications and end-users.

3.7 Conclusion

In conclusion, data replication in fog computing presents a pivotal strategy for enhancing data availability, reliability, and performance in distributed edge environments. By strategically duplicating data across fog nodes, organizations can mitigate latency, improve fault tolerance, and ensure seamless access to critical information. However, effective data replication strategies must consider factors such as network bandwidth, storage capacity, and data consistency requirements to optimize resource utilization and minimize overhead. As fog computing continues to evolve, innovative approaches to data replication will play a crucial role in maximizing the efficiency and resilience of edge computing infrastructures.

6

CHAPTER 4

STRATEGY PROPOSED

4.1 Introduction

In the evolving context of Fog Computing, effective management of replicated data is crucial for ensuring the availability, reliability, and consistency of distributed data. This chapter focuses on a key strategy to address this challenge: managing the coherence of replicated data across various Fog Computing nodes. This approach aims to optimally synchronize files among the different entities of Fog Computing, including edge devices. By ensuring this synchronization, it allows users to always access the most recent and accurate version of the data, regardless of their location within the extended Fog network.

4.2 Objective

The primary objective of this strategy is to ensure that each data copy, replicated across various nodes in the Fog network, is systematically updated with the most recent version of information. This guarantees complete and coherent synchronization of files among all entities in the Fog Computing environment. By maintaining this rigorous synchronization, the strategy aims to provide a smooth, reliable, and uninterrupted user experience, allowing users to access the most up-to-date data in real time, regardless of their location within the distributed Fog network.

4.3 Strategy Proposed

When a user wants to access a file with multiple replicas, it is imperative to ensure access to the most recent version available. This requirement may necessitate synchronization among different Fog Computing nodes to retrieve the latest version. To improve this process, we propose a strategy based on the idea that users often need to access a specific part of the file rather than the entire replica. Our approach involves dividing the replica into blocks, with each block containing a specific version. Consequently, it is not necessary to access the most recent version of the entire file, but only the required block. Thus, the user can access the latest version of the part of the file they wish to view. This approach improves data consistency in a distributed file system within the Fog Computing environment while optimizing data performance and availability. The proposed strategy focuses on the following key elements:

- **File Version Management:** Files are associated with specific versions, and their version number is incremented with each modification, whether it is the addition, deletion, or modification of blocks.
- **Block Version Management:** For each modified data block, version numbers are incremented individually, allowing their evolution to be tracked autonomously.

4.4 Request processing

4.4.1 Read requests

When a Fog node receives a read request, the process of handling these requests can be detailed as follows:

1. **Data Localization:** Ensure that data is stored on appropriate Fog nodes, ideally distributed geographically to minimize latency for end users. This strategic data distribution allows for a quick response to user requests by reducing transmission delays.
2. **Block Version Management:** Implement a block version management system to track data modifications. In a blockchain-like architecture, each new version of data results in the addition of a new block to the chain. The most recent version of the data is always stored in the latest block, enabling quick access to the latest information.
3. **Processing Read Requests:**

   (1) **Receiving the Read Request:** When a read request is received by the local Fog node, it triggers the processing to retrieve the requested data. The Fog node acts as an intermediary between the end user and the distributed storage system.

7

(2) **Consulting the Head of the Blockchain:** The local Fog node consults the head of the blockchain to identify the most recent block. The head of the blockchain corresponds to the last block added, containing the most recent data. This step is crucial to ensure that the user receives the current version of the data.

(3) **Data Retrieval:** Once the most recent block is identified, the local Fog node retrieves the data from this block. Depending on the structure of the blockchain (e.g., a tree structure), the Fog node might need to navigate through this structure to extract the specific requested data.

4. **Data Transmission to the User:** Once the data is retrieved, the local Fog node transmits it to the user who issued the read request. Data transmission must be conducted through secure communication protocols to ensure the integrity and confidentiality of the data during the transfer.

5. **End of the Request:** After transmitting the data to the user, the read request is considered complete. The local Fog node is then ready to process other requests or perform other tasks.

4.4.2 Write Requests

The process of handling write requests in a fog computing environment, particularly when updating data in a blockchain, can be described in detail as follows:

1. **Receiving the Write Request:** When a write request is received by the local Fog node, the processing begins to update the data. This initial step involves receiving and acknowledging the user's request to modify the data.

2. **Consulting the Head of the Blockchain:** The local Fog node consults the head of the blockchain to identify the most recent block, which contains the latest version of the data. The head of the blockchain is the reference point for the current state of the data.

3. **Validating the Request:** Before updating the data, the request is validated to ensure that the appropriate rules and permissions are followed. This step involves verifying access rights and compliance with security and data management policies.

4. **Updating the Data in the Existing Block:** Once the request is validated, the local Fog node proceeds to update the data directly in the most recent block. This may involve adding, modifying, or deleting existing information in this block. The update is done in a manner that ensures the integrity and consistency of the data.

5. **Incrementing the Block Version:** After updating the data in the existing block, the block version is incremented to reflect the changes made. This can be achieved by incrementing

a version number or adding a version indicator to the block itself, ensuring that the most recent version of the data is always easily identifiable.

4.5 Management of Available Blocks and Waiting States

The process begins by initializing and then verifying if a local replica of the recent data block exists. If a local replica is found, the block is accessed directly. If not, the system enters a waiting state where it periodically checks for the block's local availability. During this waiting state, if the block becomes available locally, it is accessed immediately. If the block is still not found and a timeout occurs, the system checks for the block in neighboring fogs. If a neighboring fog has the replica, the block is accessed via connection to that fog. If no replica is found in the neighboring fogs, the process finishes. This loop continues until the block is successfully accessed or the process concludes, prioritizing local access, followed by periodic rechecks, and finally attempting to access from neighboring fogs if necessary.

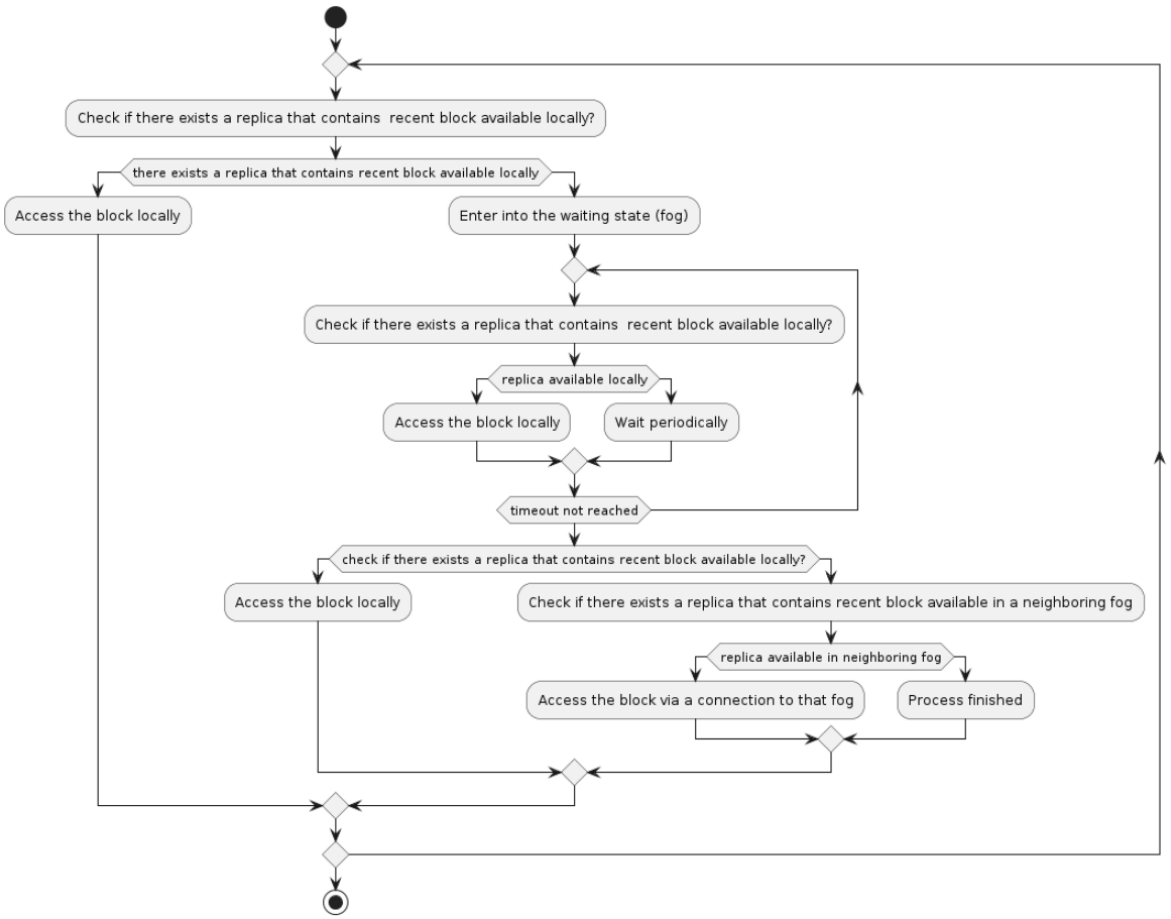

Figure 4.1: processing requests.

8

4.6 Synchronization

Synchronization between replicas containing versions in Fog Computing refers to the coordination process aimed at aligning the different versions of replicated data distributed across the nodes in the Fog network. This synchronization ensures that each replica has the latest version of the data, thereby ensuring consistency and up-to-date information throughout the distributed system. Synchronization between replicas in Fog Computing can indeed vary depending on the desired or required level of consistency for the distributed data. This consistency can be classified into several levels, generally described as strong, medium, and weak:

- **Strong Consistency:**
  "Read One Write All" (ROWA) is a data consistency model in distributed systems where

any write made to one data replica must be propagated to all other replicas before any subsequent read can occur.

- **Medium Consistency:**

  In distributed systems, medium consistency involves regular synchronization of data between replicas, thereby reducing the risk of divergence between versions.

- **Weak Consistency:**

  Weak consistency allows delays in the propagation of updates between replicas, permitting temporary differences between distributed copies. This can lead to different or outdated results when reads are performed on different replicas within a given time frame.

4.6.1 Synchronization Methods

Data synchronization of replicated data in distributed environments such as Fog Computing is essential to ensure data consistency and accessibility. Three main approaches are often used to manage this synchronization effectively.

- **Periodic Synchronization:** Implement scheduled synchronization at regular intervals or triggered by an event between replicated nodes.

- **Modification-Based Synchronization:** Define a modification threshold for each node. When a node reaches this threshold, it triggers synchronization with other nodes to disseminate the changes. This reduces the frequency of synchronizations while ensuring data is regularly updated based on node activity.

- **Hybrid Synchronization:** In this approach, both scheduled synchronizations at regular intervals and synchronizations triggered by specific events (such as reaching a modification threshold on a node) are used. The idea is to leverage the advantages of both methods: periodic synchronization maintains regular data updates, ensuring overall system consistency at predefined intervals, while modification-based synchronization responds reactively to changes. This hybrid approach provides an efficient and precise solution for determining the optimal timing of file version synchronization in a Fog environment. By considering both the frequency of modifications and an optimal synchronization period, it becomes possible to effectively determine the ideal timing for synchronization.

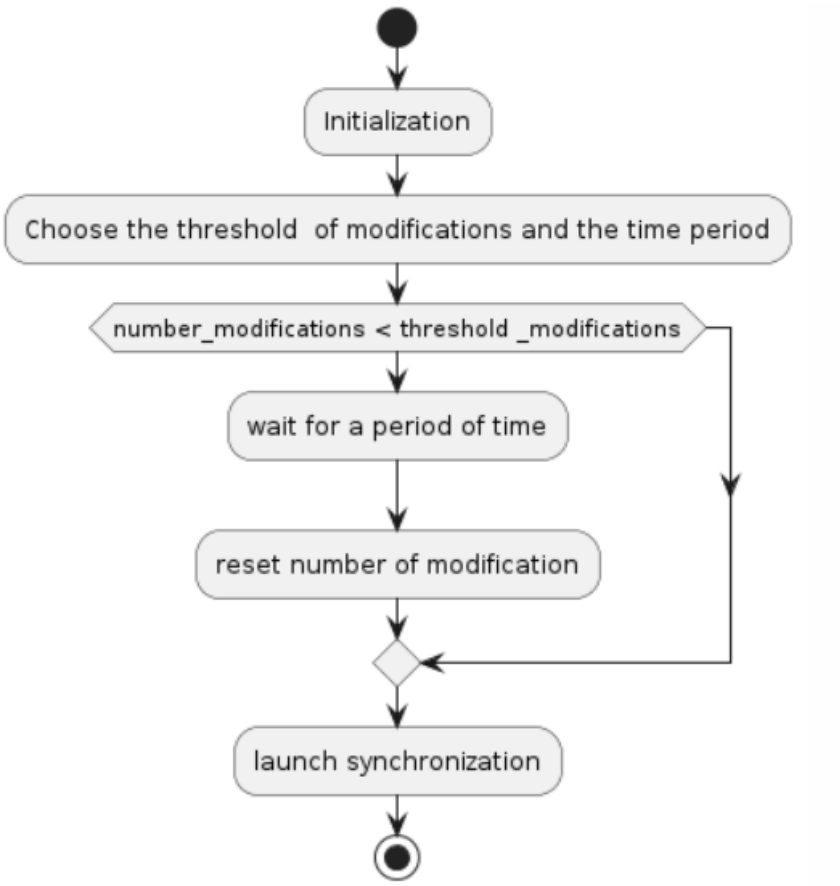

Figure 4.2: Hybrid synchronization.

9

4.6.2 Adaptive Synchronization Strategies in Fog Computing Environments

To ensure data consistency and reliability in a distributed environment, we implement a synchronization process at the level of each Fog node. Within this framework, we represent versions of different replicas in a matrix, where rows correspond to data block versions and columns to different replicas. After each read request, the block version remains unchanged, while a write request increments the block version. Our goal is to optimize this process to minimize impact on user network access. During synchronization, user access is temporarily blocked to ensure consistent and efficient data updates. With this aim, we propose two synchronization methods: centralized and distributed.

4.6.3 Centralized synchronization

In this framework of centralized synchronization, Fog nodes communicate with a central entity, often the Cloud, to ensure data consistency across the entire network. During this process, user access to Fog nodes and the Cloud is restricted, thereby ensuring that operations proceed in a controlled and secure manner.

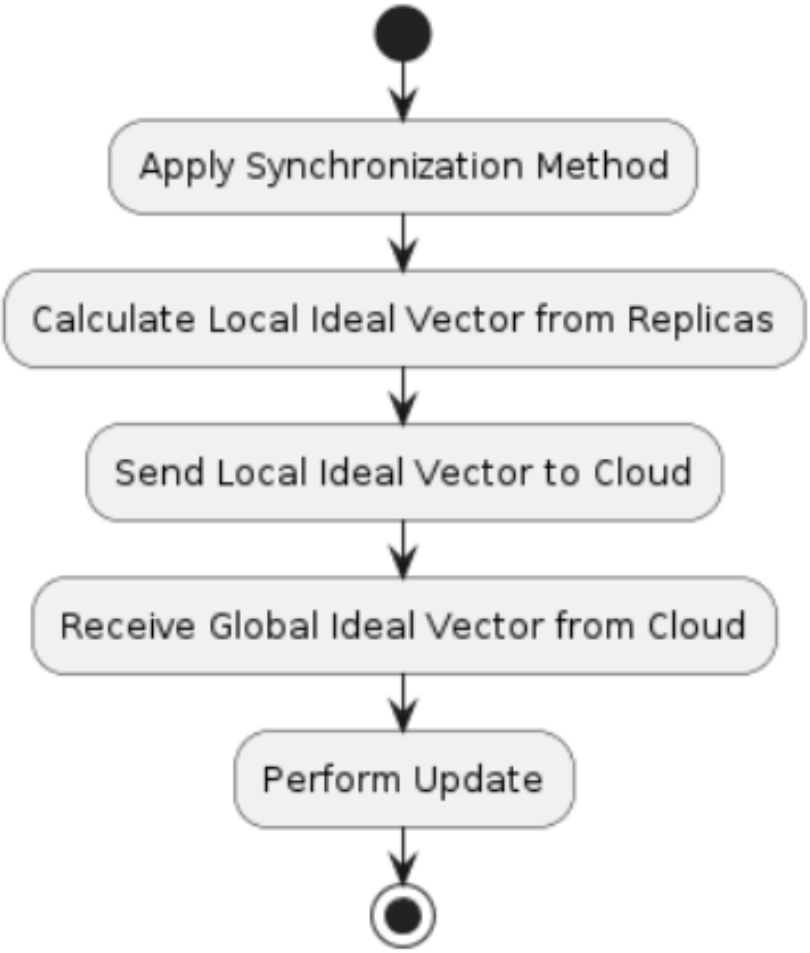

Figure 4.3: centralized synchronization in the fog.

10

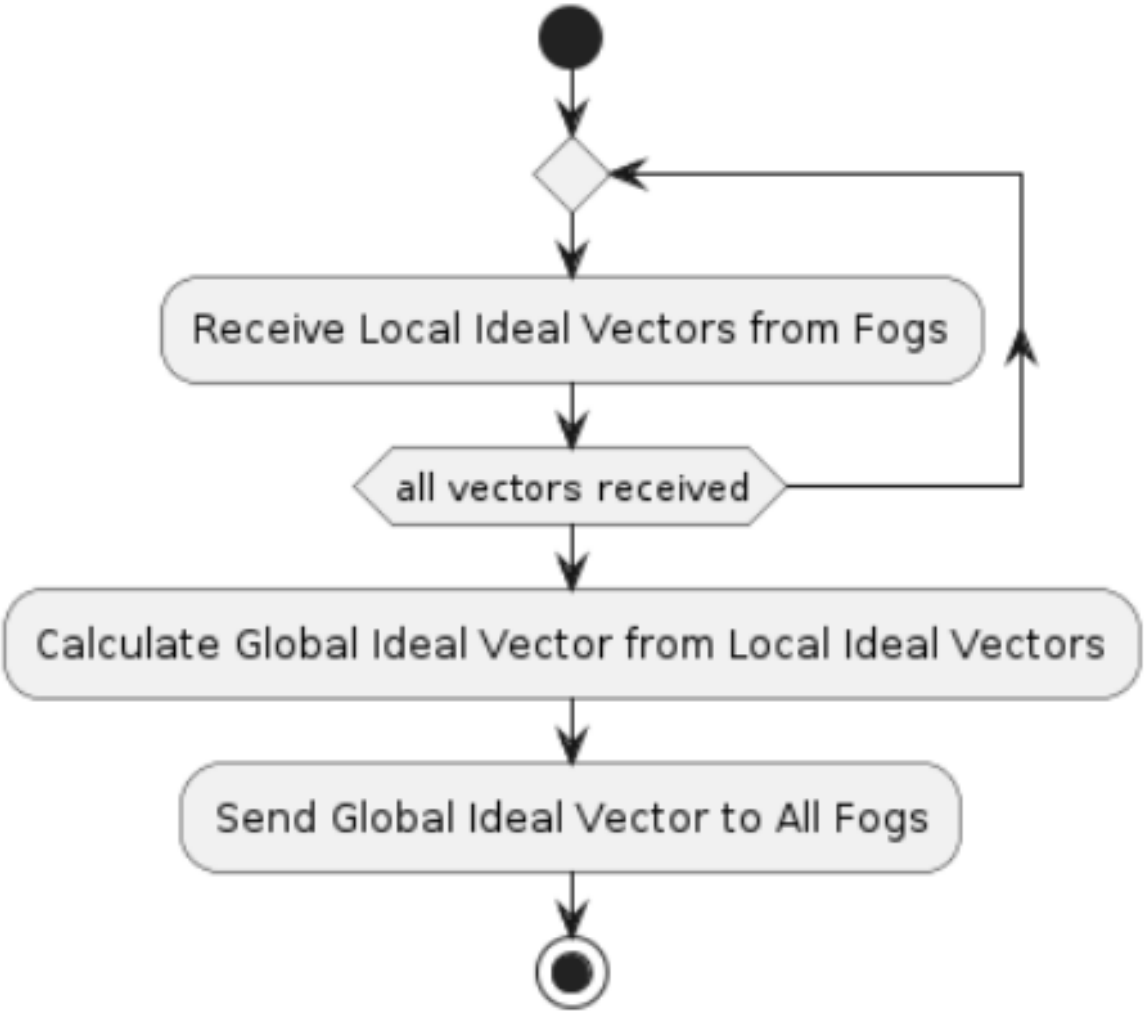

Figure 4.4: centralized synchronization in the fog.

4.6.3.1 Synchronization Triggering

- **Modification-Based Method:** When a Fog node detects that the number of modifications to a block reaches a predefined threshold, it sends its local ideal vector to the Cloud. Upon receiving this vector, the Cloud triggers a global synchronization by requesting other Fog nodes to construct and send their local ideal vectors.

- **Periodic Method:** If synchronization is scheduled to occur at regular intervals, the first Fog node to reach the predefined interval sends its local ideal vector to the Cloud. Upon receiving this vector, the Cloud initiates the same procedure, requesting other Fog nodes to construct and send their local ideal vectors.

4.6.3.2 Synchronization Process

- **Process on Fog Nodes:**
  1. **Calculation of the Local Ideal Vector:** Each Fog node calculates its local ideal vector based on the replicas it holds. These replicas contain the versions of the data blocks, thus ensuring local consistency without requiring constant communication with other nodes.
  2. **Sending the Local Ideal Vector to the Cloud:** Once calculated, each Fog node sends its local ideal vector to the Cloud.
  3. Reception of the Global Ideal Vector from the Cloud.
  4. Initiation of the Update.
- **Process on the Cloud:**
  1. **Reception of Local Ideal Vectors from Fog Nodes:** The Cloud receives the local ideal vectors from all Fog nodes. This centralization of data allows for a comprehensive analysis of the system's state, facilitating decision-making and performance optimization.
  2. **Calculation of the Global Ideal Vector:** Using the received local ideal vectors, the Cloud calculates a global ideal vector. This ensures a consolidated and accurate view of the system's state, which can help in detecting inconsistencies and anomalies.
  3. **Sending the Global Ideal Vector to Fog Nodes:** Once calculated, the global ideal vector is sent to all Fog nodes.
- **Advantages:**
  1. **Global Data Consistency:** By centralizing the synchronization process around a central entity, typically the Cloud, centralized synchronization ensures global data consistency across the entire network.
  2. **Ease of Management:** By concentrating the coordination and management of data within the Cloud, centralized synchronization simplifies the supervision and administration of the system, thereby reducing operational complexity.
- **Disadvantages:**
  1. **Potential Latency:** Data exchanges between Fog nodes and the Cloud can cause delays, especially in environments where latency is crucial. This situation can negatively impact real-time performance and system responsiveness.
  2. **Bandwidth Requirements:** Frequent communication between Fog nodes and the Cloud requires adequate bandwidth, which can become an obstacle in large-scale networks or environments where connectivity is limited.

4.6.4 Distributed synchronization

In distributed synchronization, Fog nodes communicate directly with each other to avoid issues posed by centralized synchronization, such as latency and bandwidth consumption. This process involves temporarily blocking user access to ensure data consistency. Here's how it unfolds:

- **Application of Synchronization Method:**
  Each Fog node applies an appropriate synchronization method, either based on a periodic time interval or triggered by a certain number of modifications.
- **Construction of the Matrix:**
  During this process, each Fog node constructs a matrix based on the replicas and versions of the data blocks it holds. This matrix reflects the local state of the data.
- **Calculation of the Local Ideal Vector:**
  After constructing the matrix, each Fog node calculates a local ideal vector, representing the optimal state of the data based on available local information.
- **Exchange of Vectors between Fog Nodes:**
  Each Fog node sends its local ideal vector to all other Fog nodes in the network. This exchange allows each node to share its local state with others.
- **Construction of the Global Ideal Vector:**
  When a Fog node receives local ideal vectors from (n-1) other nodes, it uses this information to construct a global ideal vector, reflecting a consistent view of the data across all nodes.
- **Completeness Test and Update:**
  Each Fog node waits to receive all ideal vectors from other nodes. Once all vectors are received, the node checks completeness and, if all conditions are met, updates its data to align with the global ideal vector.

  This approach to distributed synchronization enhances responsiveness and reduces bandwidth requirements by eliminating dependence on a central entity. It also allows for faster and more efficient data updates across the network of Fog nodes.

4.7 Example

In this example, we consider three Fog environments, each associated with a matrix based on replicas and versions of data blocks. Each Fog environment executes a specific process to produce a local ideal vector, representing the most recent version of data blocks for each row. This vector is then sent to all other Fog nodes. When a Fog node receives local ideal vectors from other nodes (n-1 vectors, where n represents the total number of Fog nodes), it calculates a global ideal vector

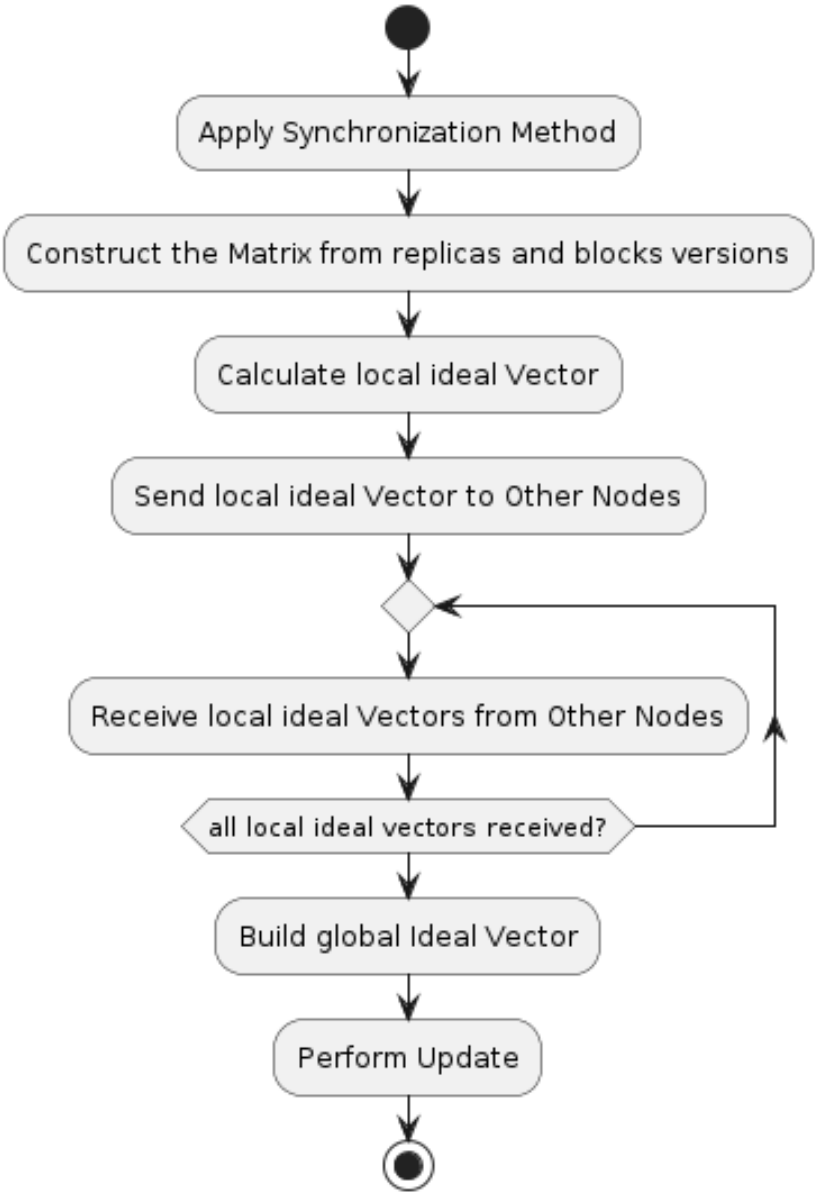

Figure 4.5: Distributed synchronization.

11

by comparing the local ideal vectors and selecting the most recent version of data blocks. Here are the initial matrices for each Fog environment:

Fog1

$Row1 : [25,5,9]$
$Row2 : [2,4,8]$
$Row3 : [1,6,7]$

Fog2

$Row1 : [11,13]$
$Row2 : [10,18]$
$Row3 : [16,17]$

Fog3

$Row1 : [11,13,15]$
$Row2 : [12,14,12]$
$Row3 : [26,27,29]$

Process of calculating the local ideal vector for each Fog:

- **Fog 1:**
  - Row 1: [25,5,9] (sorted) => Most recent version: 25
  - Row 2: [2,4,8] (sorted) => Most recent version: 8
  - Row 3: [1,6,7] (sorted) => Most recent version: 7
  - Local Ideal Vector: [25,8,7]
- **Fog 2:**
  - Row 1: [11,13] (sorted) => Most recent version: 13
  - Row 2: [10,18] (sorted) => Most recent version: 18

12

  - Row 3: [16,17] (sorted) => Most recent version: 17
  - Local Ideal Vector: [13,18,17]

- **Fog 3:**
  - Row 1: [11,13,15] (sorted) => Most recent version: 15
  - Row 2: [12,14,12] (sorted) => Most recent version: 14
  - Row 3: [26,27,29] (sorted) => Most recent version: 29
  - Local Ideal Vector: [15,14,29]

- **Local Ideal Vectors:**
  - Local Ideal Vector of Fog 1: [25,8,7]
  - Local Ideal Vector of Fog 2: [13,12,17]
  - Local Ideal Vector of Fog 3: [15,18,29]

- **Calculation of the Global Ideal Vector:** By comparing the local ideal vectors and selecting the most recent versions for each element:
  - Global Ideal Vector: [25,18,29]

- **Updated Matrices for each Fog:**
  - **fog 1:**
    [25,25,25]
    [18,18,18]
    [29,29,29]
  - **fog 2:**
    [25,25]
    [18,18]
    [29,29]
  - **fog 3:** [25,25,25]
    [18,18,18]
    [29,29,29]

4.8 Optimization of Distributed Synchronization with the Vague Algorithm

To improve distributed synchronization and minimize complexity, we have used the distributed Vague algorithm. The Vague algorithm offers an efficient solution for managing distributed systems, particularly in synchronization of updates. Its main advantages include optimized data dissemination across the network, significant latency reduction through hierarchical message management, efficient bandwidth usage by limiting data exchanges, increased robustness against failures, and adaptability to the different needs of distributed applications. Overall, it enhances the performance and reliability of distributed systems while maintaining precise and consistent data synchronization.

4.9 Integration of Distributed Synchronization with the Vague Algorithm

To further enhance the efficiency and robustness of distributed synchronization, we integrate the distributed synchronization process with the Vague algorithm. This combined approach leverages the hierarchical and efficient data dissemination capabilities of the Vague algorithm while maintaining the detailed steps of distributed synchronization. Here's how the integration works:

- **Vague Algorithm Structure:**
  The Vague algorithm structures the network of Fog nodes into a hierarchical system. Each level in the hierarchy has nodes that manage synchronization for the nodes directly below them, creating a multi-tiered network.
- **Triggered Synchronization Methods:**
  Each Fog node at the lowest level of the hierarchy applies a synchronization method based on a periodic time interval or triggered by a certain number of modifications.
- **Construction of the Local Matrix:**
  Each Fog node constructs a matrix reflecting the local state of the data based on the replicas and versions of the data blocks it holds.
- **Calculation of the Local Ideal Vector:**
  After constructing the matrix, each Fog node calculates a local ideal vector, representing the optimal state of the data based on available local information.
- **Hierarchical Exchange of Vectors**
  Instead of each Fog node sending its local ideal vector to all other nodes, the exchange is managed hierarchically:

13

  - Local ideal vectors are first sent to the immediate upper-level nodes in the hierarchy.
  - Upper-level nodes aggregate the vectors and pass them further up the hierarchy until they reach the top level.
  - The top-level node constructs a global ideal vector and disseminates it back down the hierarchy.

- **Construction of the Global Ideal Vector:**
  Each top-level node constructs the global ideal vector using the aggregated information from all lower-level nodes. This vector reflects a consistent view of the data across the entire network.

- **Dissemination and Update:**

  - The global ideal vector is disseminated back down the hierarchy, with each level ensuring that all nodes below them receive the updated vector.
  - Each Fog node, upon receiving the global ideal vector, checks for completeness and updates its data to align with the global ideal vector.

By integrating the distributed synchronization process with the Vague algorithm, we achieve a more efficient, robust, and scalable synchronization mechanism for Fog Computing environments, enhancing overall performance and user experience.

4.10 Conclusion

In this chapter, we have detailed our strategy for ensuring data consistency in a Fog Computing environment. The primary objective of this strategy is to ensure that each user always accesses the most recent version of the files. We began by describing the process that allows the user to efficiently access the latest available version. To ensure data consistency, we implemented rigorous synchronization based on various methods. We presented two approaches for managing data synchronization: centralized and distributed. Furthermore, we proposed an improvement to the distributed strategy by integrating the vague algorithm. This approach optimizes synchronization and the dissemination of updates in Fog Computing environments, thus offering improved performance in terms of latency reduction, efficient bandwidth usage, and system robustness. In summary, our strategy ensures reliable and consistent data management across all Fog entities, resulting in an optimized user experience.

CHAPTER 5

SIMULATION

5.1 Introduction

In the previous chapter, we presented our strategy for managing the coherence of replicated data in Fog computing and detailed its functioning. This chapter is dedicated to the implementation phase of this strategy with the aim of evaluating its performance and validating the proposed strategy. To achieve this, we extended the IFogSim simulator, which is developed in Java. We conducted several series of experiments, the results and interpretations of which are discussed in this chapter.

5.2 IFogSim simulator

IFogSim is a simulator specifically designed to model and analyze Fog Computing and Internet of Things (IoT) environments, iFogSim allows researchers and developers to simulate various resource management policies, task allocation, and data processing to optimize the performance of IoT applications. By providing a detailed platform to evaluate the impacts of latency, energy consumption, and bandwidth usage, iFogSim facilitates academic research, the development of IoT solutions, and the teaching of Fog Computing concepts. iFogSim is a Java based open-source simulation tool for simulating fog computing scenarios. It is developed by Harshit Gupta and the team at the Cloud Computing and Distributed Systems (CLOUDS) Lab University of Melbourne Australia [7].

Language used: Java.
IDE used: Eclipse.

5.3 Java programming language

Java is an object-oriented programming language created by James Gosling and Patrick Naughton, employees of Sun Microsystems, with the support of Bill Joy (co-founder of Sun Microsystems in 1982). It was officially introduced on May 23, 1995, at SunWorld. Sun Microsystems was later acquired by Oracle Corporation in 2009, which now owns and maintains Java [15]. The primary feature of the Java language is that software written in Java can be easily portable across multiple operating systems such as UNIX, Windows, Mac OS, or GNU/Linux, with or without modifications. The platform guarantees the portability of applications developed in Java [15].
Today, Java has become an indispensable direction in the world of programming [15].

5.4 Development environment

Eclipse is an integrated development environment (IDE) released as open-source by Sun in June 2000. In addition to Java, Eclipse supports various other languages such as Python, C, C++, JavaScript, XML, Ruby, PHP, and HTML. It is available for download at `https://www.eclipse.org/downloads`. Eclipse includes all the features of a modern IDE (color-coded editor, multi-language projects, refactoring, graphical interface editor, and web page editor). Designed in Java, Eclipse is available on Windows, Linux, Solaris, Mac OS, or as a platform-independent version (requiring a Java virtual machine). Furthermore, Eclipse is open-source and can be downloaded directly from `http://java.sun.com`. It is powerful and compatible with all new Java technologies (Java EE technologies, databases, UML, XML, etc.).

5.5 Simulation configuration

Before any simulation, the first action to perform is the configuration of the simulated infrastructure. We implemented a control panel.

5.6 Launching simulations and viewing results

To highlight the contributions of our approaches, we will focus on the following metrics: execution time, energy consumed, and network usage. To study the behavior of our proposals and analyze the results obtained from the simulation, we will compare both strategies: centralized and distributed. Several series of simulations were launched.

14

5.6.1 Simulation 1

In this simulation, we created 10 IoT devices and set the number of replicas per fog node to 4. We then varied the number of fog nodes from 2 to 8. Our study focused on examining the impact of these variations on three critical metrics: execution time, energy consumption, and network usage, to gain a comprehensive understanding of system efficiency under different conditions.

5.6.1.1 Execution time

fog number	2	3	4	5	6	7	8
distributed strategy	84	215	369	149	397	298	438
centralized strategy	108	218	190	212	366	366	503

Table 5.1: Impact of the number of fogs on the execution time.

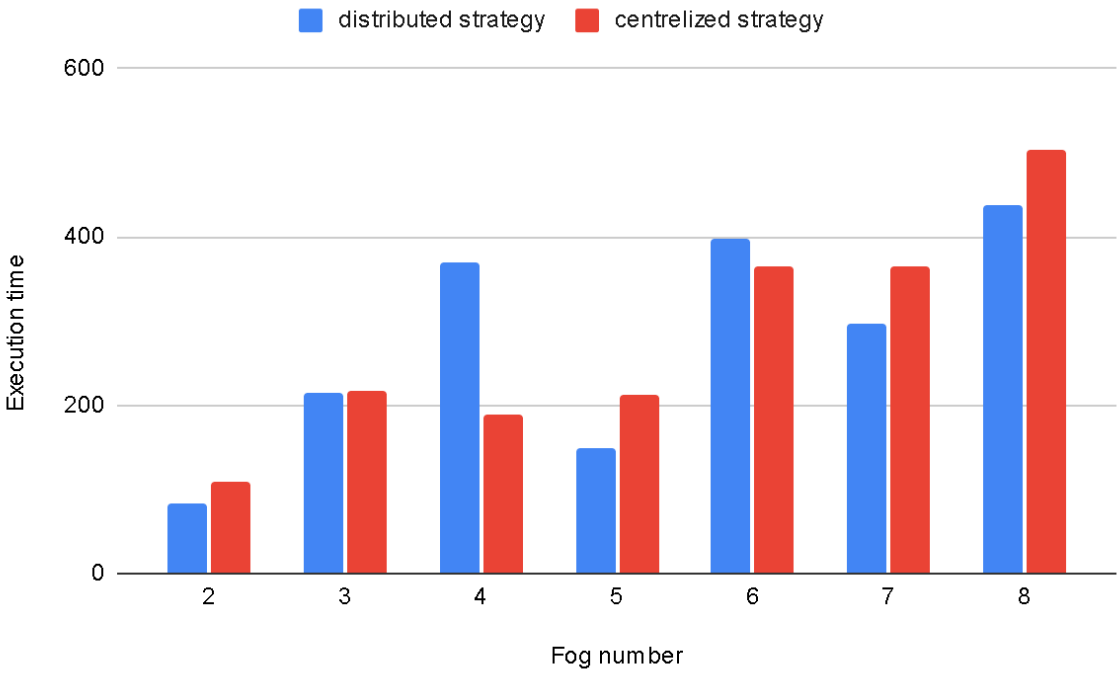

Figure 5.1: Impact of the number of fogs on the execution time.

fog number	2	3	4	5	6	7	8
distributed strategy	1000518,395	752071,5153	667003,6737	556116,823	476806,2251	417552,0022	355762,452
centralized strategy	1003318,024	796827,2479	670843,2635	596499,6726	471912,2581	392407,9319	367262,2001

Table 5.2: Impact of the number of fogs on the energy consumed.

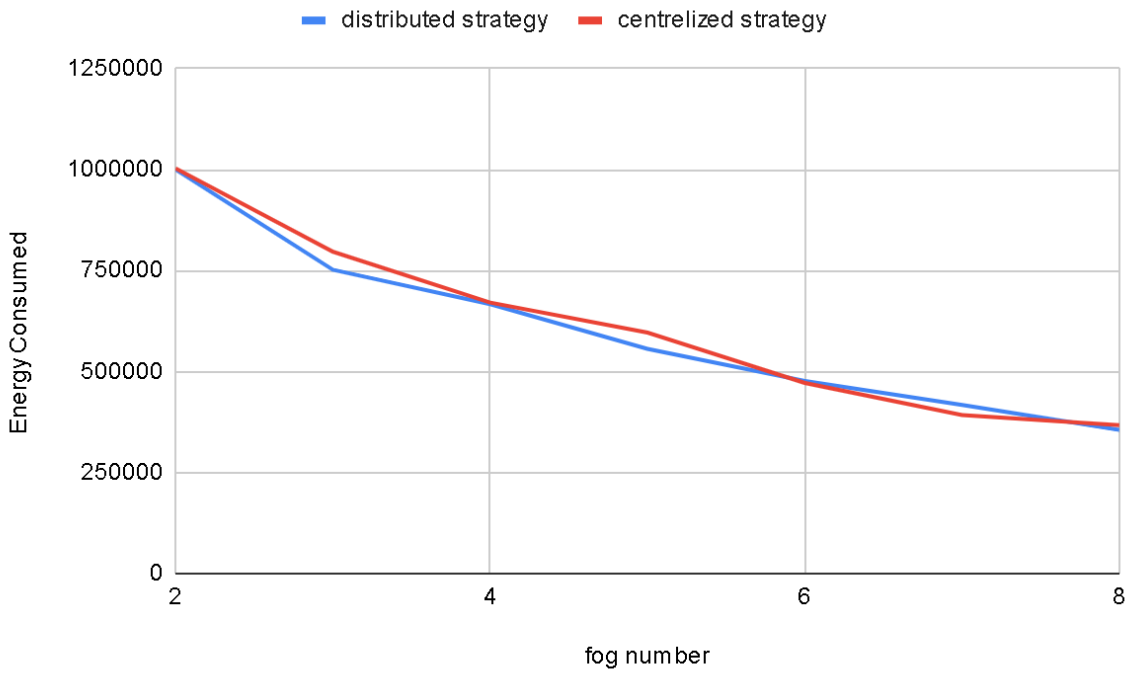

Figure 5.2: Impact of the number of fogs on the energy consumed.

5.6.1.2 Energy consumed

5.6.1.3 Network usage

fog number	2	3	4	5	6	7	8
distributed strategy	20	34,5	40	44,5	54,5	75	80
centralized strategy	25	59,5	75	109,5	154,5	114,5	155

Table 5.3: Impact of the number of fogs on the network usage.

15

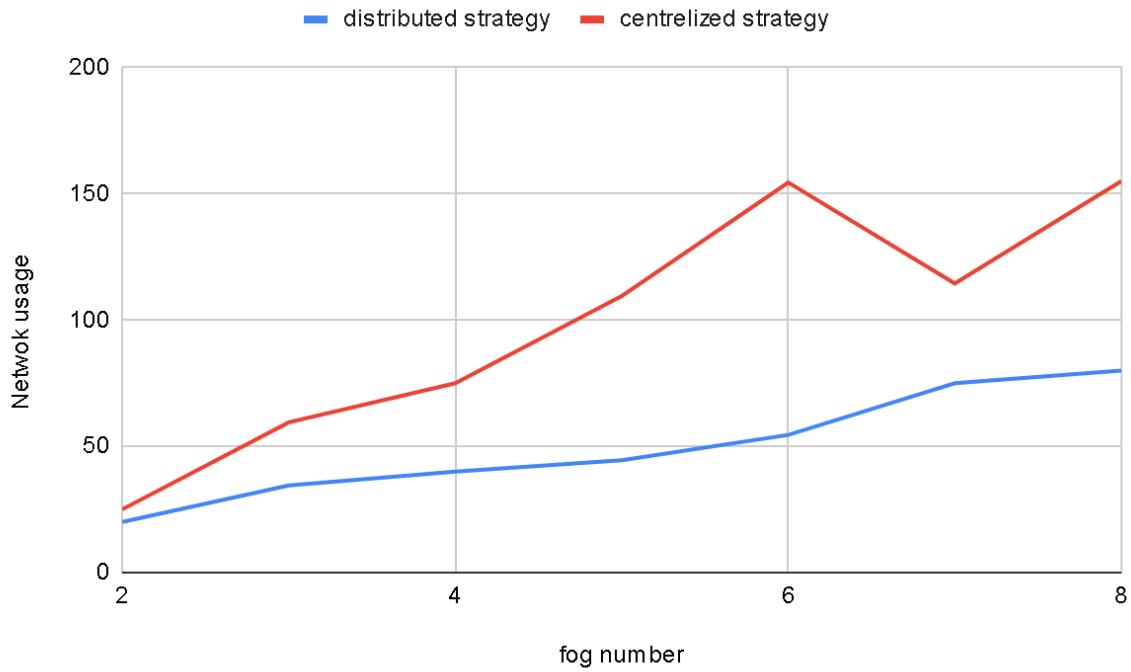

Figure 5.3: Impact of the number of fogs on the network usage.

5.6.1.4 General Observations

The analysis demonstrates that the distributed strategy significantly outperforms the centralized strategy in terms of execution time, energy consumption, and network usage. The distributed strategy achieves lower execution times and reduced network usage, with decreasing energy consumption as the number of fog nodes increases, indicating greater overall efficiency. In contrast, the centralized strategy consistently exhibits higher execution times, energy consumption, and network usage. Regarding latency, the centralized strategy requires communication with a central entity, often the Cloud, which introduces significant delays due to the round-trip time needed for data transmission. Conversely, the distributed strategy allows nodes to communicate directly, reducing transmission distance and time, resulting in faster synchronization and more responsive systems. In terms of bandwidth consumption, the centralized strategy involves sending data to and from the Cloud, consuming substantial bandwidth, especially with large data blocks or high network activity. The distributed strategy reduces overall bandwidth usage by enabling direct data sharing between Fog nodes, avoiding the need for a central data route. Energy efficiency also favors the distributed strategy. The centralized approach's constant communication with the Cloud increases energy consumption due to the need to maintain connections and handle

large data volumes. The distributed strategy, however, manages synchronization locally, leading to reduced energy consumption through less data transmission over long distances and fewer connections to maintain.
Overall, the distributed strategy is more efficient and superior.

5.6.2 Simulation 2

In this simulation, we created 2 nodes of fog and set the number of replicas per fog node to 4. Then we varied the number of IoT devices from 10 to 100, to analyze the three metrics.

5.6.2.1 Execution time

Number of IOT devices	10	20	30	40	50	60	70	80	90	100
Distributed strategy	190	132	299	155	141	165	232	276	244	278
Centralized strategy	329	645	528	503	251	451	383	462	389	418

Table 5.4: Impact of number of IOT devices on the execution time.

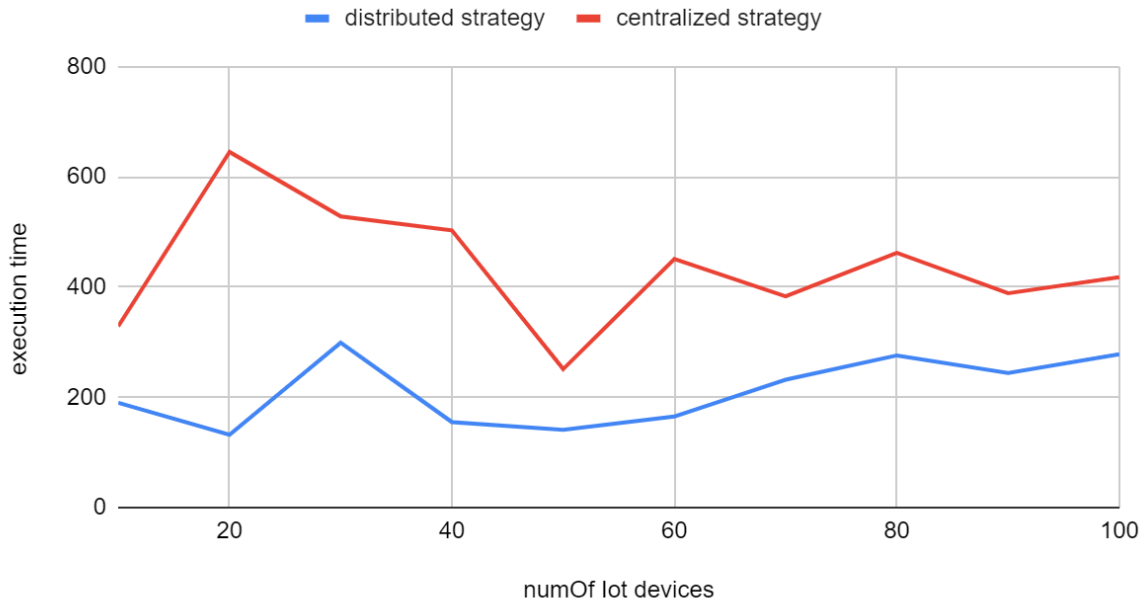

Figure 5.4: Impact of number of IOT devices on the execution time.

16

5.6.2.2 Energy Consumed

Number of IOT devices	10	20	30	40	50	60	70	80	90	100
Distributed strategy	1003165.795	1000851.046	1000631.542	1007443.354	997911.0667	997911.0667	997911.0667	997911.0667	997911.0667	999844.4
Centralized strategy	1003045.488	1004267.617	1008245.664	1007441.838	999244.4	998244.4	998244.4	998244.4	998244.4	997911.07

Table 5.5: Impact of number of IOT devices on the Energy Consumed.

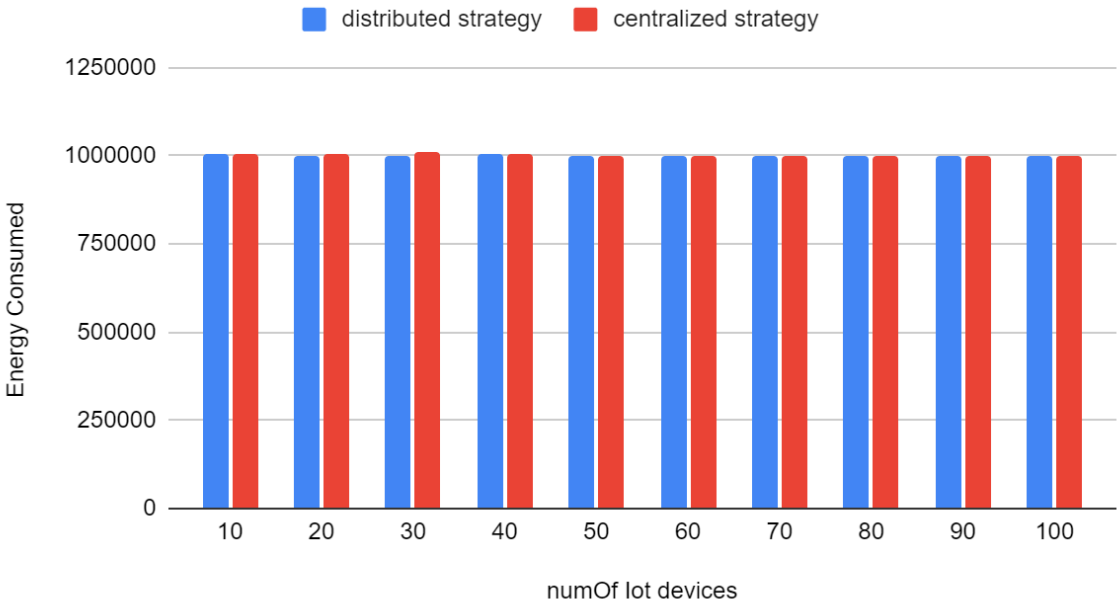

Figure 5.5: Impact of the number of IOT devices on the Energy Consumed.

Number of IOT devices	10	20	30	40	50	60	70	80	90	100
Distributed strategy	30	70	165	120	150	180	208.5	240	265.5	297
Centralized strategy	35	230	675	118.5	147	178.5	210	383	265.5	298.5

Table 5.6: Impact of the number of IOT devices on the Network Usage.

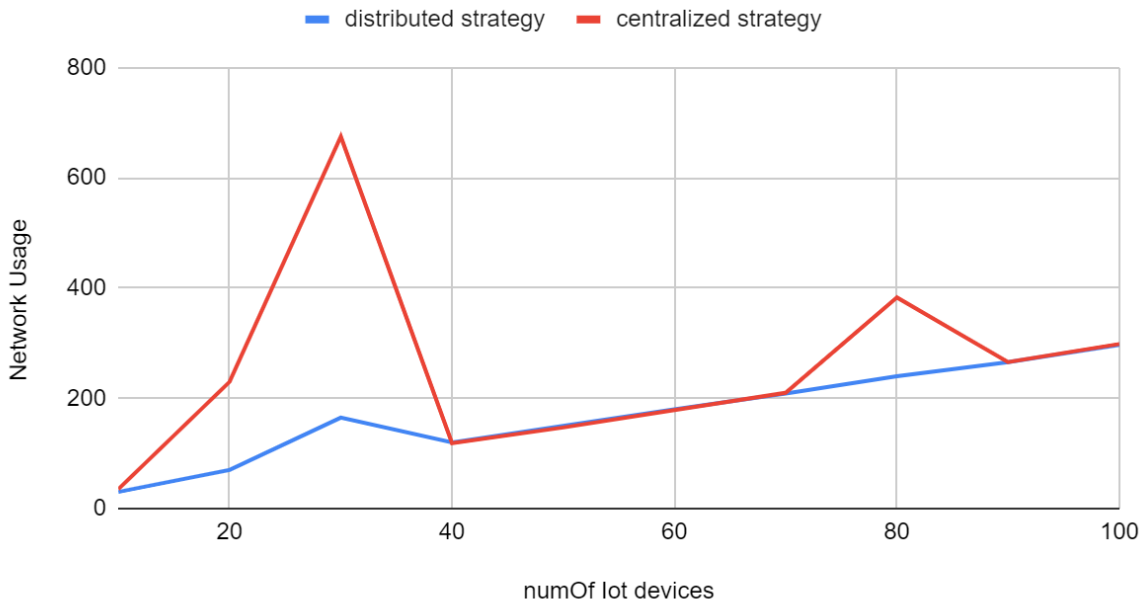

Figure 5.6: Impact of the number of IOT devices on the Network Usage.

5.6.2.3 General Observations

- **Execution Time:** The distributed strategy performs better with lower execution times compared to the centralized strategy as the number of IoT devices increases.

- **Energy Consumed:** Both strategies show a similar trend of decreasing and stabilizing energy consumption with the increase in IoT devices, with the centralized strategy showing a slightly more consistent trend.

- **Network Usage:** The distributed strategy shows lower network usage compared to the centralized strategy, although both strategies see an increase in network usage as the number of IoT devices increases.

These observations suggest that the distributed strategy generally performs better in terms of execution time and network usage, while both strategies have a similar performance in terms of energy consumption.

17

5.6.3 Simulation 3

Our study aimed to evaluate the impact of three critical metrics: execution time, energy consumption, and network usage. This comprehensive analysis helps us understand system efficiency under varying conditions. In this simulation, we created 2 nodes of fog and set the number of IOT devices to 10 , Then we varied the number of replicas from (‘,8,12,16,20,24,28), to analyze the three metrics.

Number of replicas	4	8	12	16	20	24	28
Distributed strategy	204	217	292	155	242	320	196
Centralized strategy	280	145	233	316	301	166	316

Table 5.7: Impact of number of replicas on the execution time .

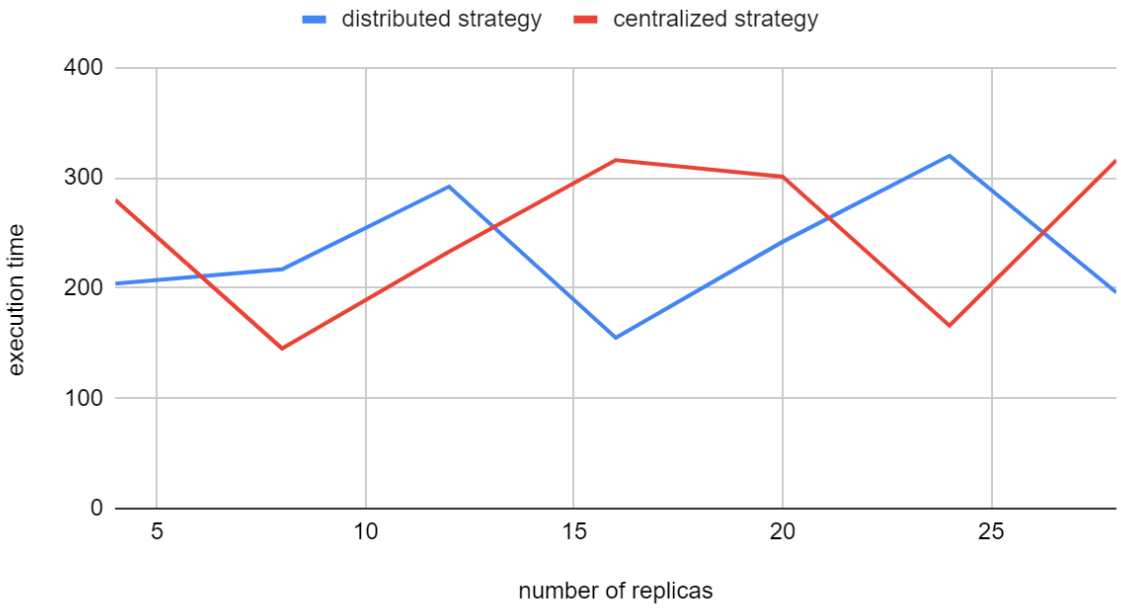

Figure 5.7: Impact of number of replicas on the execution time.

5.6.3.1 Energy consumed

Number of replicas	4	8	12	16	20	24	28
Distributed strategy	1000845.202	1001741.212	1000133.215	1000177.152	1001487.222	1000789.919	1000789.919
Centralized strategy	1000247.832	1000600.657	1001820.388	1000603.038	1002586.767	1002311.9	1002500.009

Table 5.8: Impact of number of replicas on the energy consumed.

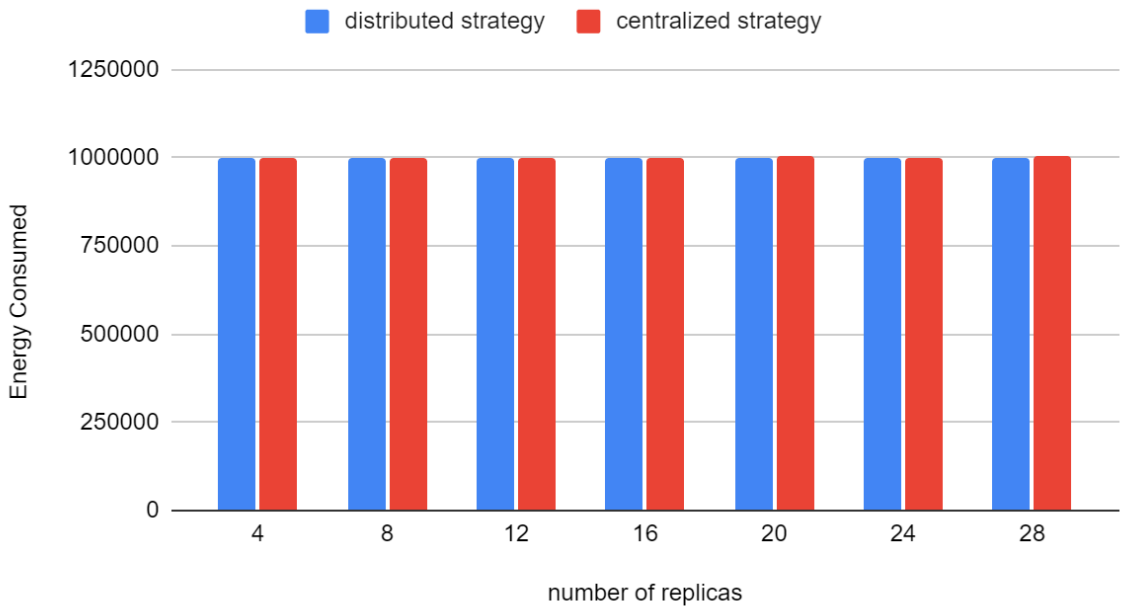

Figure 5.8: Impact of number of replicas on the energy consumed.

5.6.3.2 Network usage

Number of replicas	4	8	12	16	20	24	28
Distributed strategy	30	39.5	15	15	15	25	10
Centralized strategy	20	40	70	30	70	55	15

Table 5.9: Impact of number of replicas on the network usage.

18

Figure 5.9: Impact of number of replicas on the network usage.

5.6.3.3 General Observations

- **Execution Time:** The distributed strategy generally performs better (lower execution time) than the centralized strategy as the number of replicas increases.
- **Energy Consumed:** The distributed strategy tends to consume more energy compared to the centralized strategy, but the difference reduces with an increasing number of replicas.
- **Network Usage:** The distributed strategy starts with higher network usage but becomes more efficient (lower network usage) with more replicas, whereas the centralized strategy has a fluctuating network usage trend.

5.6.4 Simulation 4

In this simulation, we created 2 nodes of fog and set the number of IOT devices to 10 and number of replicas to 4 , Then we varied the threshold of modification from 3 to 21 to analyze its impact on three key performance metrics. This study focused on understanding how these variations affect execution time, energy consumption, and network usage.

5.6.4.1 Execution time

Threshold Of Modification	3	6	9	12	15	18	21
Distributed strategy	139	79	92	82	154	119	82
Centralized strategy	399	156	162	201	140	87	153

Table 5.10: Impact of threshold of modification on the execution time.

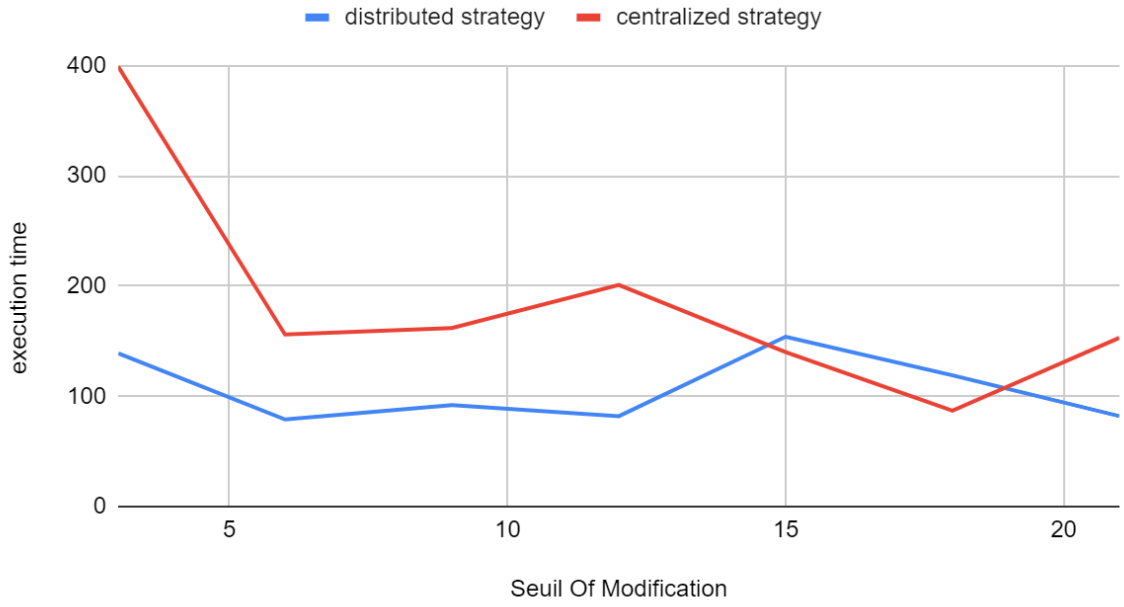

Figure 5.10: Impact of threshold of modification on the execution time .

5.6.4.2 Energy consumed

Threshold Of Modification	3	6	9	12	15	18	21
Distributed strategy	1000843.905	1000842.062	1000852.404	1000812.422	999167.1792	997491.9805	998495.1582
Centralized strategy	1002633.878	1001570.896	1001447.749	1000519.221	1000485.652	1000819.806	1000846.158

Table 5.11: Impact of threshold of modification on the energy consumed.

19

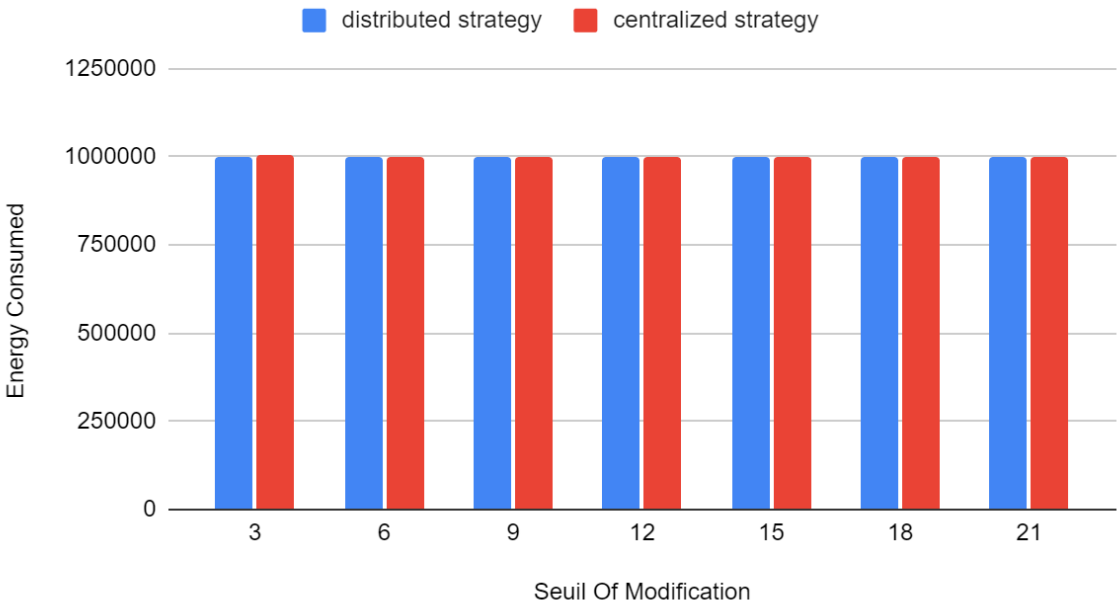

Figure 5.11: Impact of threshold of modification on the energy consumed.

5.6.4.3 Network usage

Threshold Of Modification	3	6	9	12	15	18	21
Distributed strategy	25	15	30	12	20	25	30
Centralized strategy	25	50	55	25	20	10	15

Table 5.12: Impact of threshold of modification on the network usage.

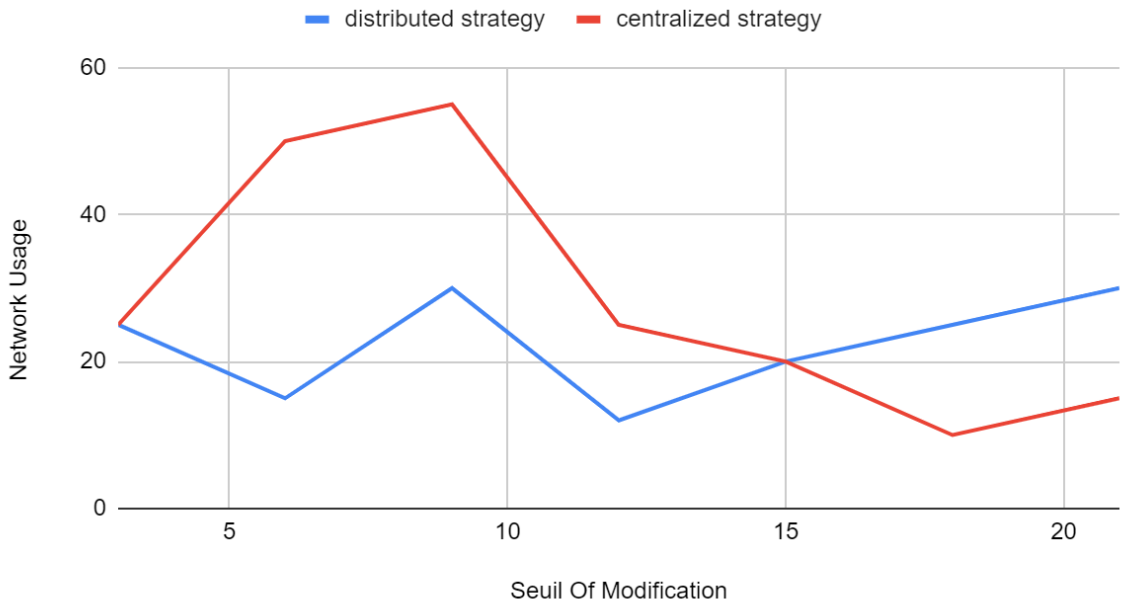

Figure 5.12: Impact of threshold of modification on the network usage.

5.6.4.4 General Observations

- **Execution Time:** The distributed strategy performs better with lower execution times across most threshold of modification.
- **Energy Consumed:** Both strategies have similar energy consumption patterns, though the distributed strategy is slightly more stable.
- **Network Usage:** The distributed strategy is generally more consistent, while the centralized strategy fluctuates more significantly.

These observations suggest that the distributed strategy might be more efficient in terms of execution time and network usage stability, while both strategies consume a similar amount of energy.

5.7 Conclusion

In this chapter, we presented the implementation of our application and the results obtained. Additionally, we conducted several series of simulations to compare the two strategies: distributed

and centralized, while varying different metrics such as execution time, energy consumed, and network usage. The comparison results showed that the **Distributed Strategy** outperforms the **Centralized Strategy** in terms of execution time and network usage, making it more efficient and superior overall. While the centralized strategy shows a slight edge in terms of energy consumption consistency, the distributed strategy's benefits in execution time and network usage make it the better choice for handling larger numbers of IoT devices or replicas.

General conclusion

Our study highlighted the transformative potential of fog computing as it brought data processing closer to the network edge, resulting in enhanced performance marked by faster processing, reduced latency, and improved resilience to network disruptions. Despite facing challenges in resource management, data consistency, and security, fog computing demonstrated its advantage by strategically replicating data across distributed nodes. This approach significantly strengthened data availability, reliability, and overall system performance by mitigating latency and enhancing fault tolerance.

Our research focused on developing a robust strategy for maintaining data consistency within fog computing environments, ensuring seamless access to the latest file versions through meticulous synchronization methods. To ensure that users accessed the latest files through rigorous synchronization methods, our approach involved dividing the replica into blocks, with each block containing a specific version. We proposed two solutions : centralized and distributed.In the centralized approach, each fog node calculated its local ideal vector based on its data replicas, ensuring local consistency. These vectors were sent to the cloud, which then calculated a global ideal vector to detect inconsistencies and anomalies. The global vector was sent back to all fog nodes for synchronization, triggered by the number of modifications, a periodic method, or a hybrid method. In the distributed approach, fog nodes communicated directly to avoid centralized latency and bandwidth issues. Each node exchanges vectors with other nodes, and constructs a global ideal vector to ensure data consistency across the system. We enhanced the IFogSim simulator to evaluate our strategies. Our experiments showed that distributed approaches generally outperform centralized ones in execution time and network usage, despite consistent energy consumption. These results validate the effectiveness of our approach, significantly enhancing fog computing's capabilities in optimizing distributed infrastructures. Looking ahead, we aim to optimize synchronization time by reducing update operations. After constructing the ideal vector following the previously described process, we proceed to compare each block in each matrix with the corresponding values of the ideal vector. Only blocks that do not match the ideal vector values will be updated, allowing for more efficient synchronization and reduced time requirements. By minimizing update time, we improve availability. Additionally, we propose a

hybrid synchronization approach combining local and global synchronization to avoid prolonged user access interruptions. Each fog environment performs local synchronization using a hybrid method (periodic and modification-based), ensuring that only the fog node reaching a specified period or modification threshold is blocked, while others continue functioning normally. During local synchronization, a local ideal vector is constructed, containing the most recent block of each replica. Even if some fog nodes have not reached the specified period or modification threshold, global synchronization is initiated using the periodic method. The local ideal vectors are then compared to form a global ideal vector, representing the most recent blocks. This global ideal vector is used for updates, ensuring efficient and consistent data synchronization across all fog environments while minimizing user service interruptions.

21

BIBLIOGRAPHY

[1] J. A.DAVIS AND W. WEIHL, *Edgecomputing: Extending enterprise applications to the edge of the internet*, 13th international World Wide Web conference, New York, (2004), pp. 180–187.

[2] H. F. ATLAM, R. J. WALTERS, AND G. B. WILLS, *Fog computing and the internet of things: A review*, big data and cognitive computing, 2 (2018), p. 10.

[3] S. DELFIN, N. SIVASANKER, N. RAJ, AND A. ANAND, *Fog computing: A new era of cloud computing*, in 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC), IEEE, 2019, pp. 1106–1111.

[4] D.KOVACHEV, *Mobile multimedia services in the cloud*, Ph.D. dissertation, RWTH Aachen University, Aachen, Germany, 2014.

[5] J. F.BONOMI, R.MILITO AND S.ADDEPALLI, *Fog computing and its role in the internet of things*, ACM SIGCOMM Workshop on Mobile cloud Computing, Helsinki, Finland, (2012), pp. 13–16.

[6] B. GHALEM, *Contribution à la gestion de la cohérence de répliques de fichiers dans les systèmes à large échelle*, (2007).

[7] H. GUPTA, A. VAHID DASTJERDI, S. K. GHOSH, AND R. BUYYA, *ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments*, Software: Practice and Experience, 47 (2017), pp. 1275–1296.

[8] C. Y. C. C. Y. W. S. M. T. C. C. C. H. S. C. C. S. J.K. ZAO, T.T. GAN AND T. JUNG, *Pervasive brain monitoring and data sharing based on multi-tier distributed computing and linked data technology*, rontiers in Human Neuroscience, 8 (2014), pp. 1–16.

[9] E.-M. K.HASHIZUME, D.G. ROSADO AND E. FERNANDEZ, *An analysis of security issues for cloud computing*, Journal of Internet Services and Applications, 4 (2013), pp. 1–13.

[10] V. KUMAR, A. A. LAGHARI, S. KARIM, M. SHAKIR, AND A. A. BROHI, *Comparison of fog computing & cloud computing*, Int. J. Math. Sci. Comput, 1 (2019), pp. 31–41.

[11] ——, *Comparison of fog computing & cloud computing*, Int. J. Math. Sci. Comput, 1 (2019), pp. 31–41.

[12] S. M.M. ISLAM AND P.GOSWAMI, *Cloud computing: A survey on its limitations and potential solutions*, International Journal of Computer Science Issues, 10 (2013), pp. 159–163.

[13] L. MOINE, *La Gestion et la sécurité dans une architecture de ressources de calcul distribuées sur l'internet*, PhD thesis, Éditeur inconnu, 2002.

[14] R. M.SATYANARAYANAN, P.BAHL AND N.DAVIES, *The case for vm-based cloudlets in mobile computing*, IEEE Pervasive Computing, 8 (2009), pp. 14–23.

[15] A. POTTS AND D. H. FRIEDEL, *Java programming language handbook*, The Coriolis Grup, 1996.

[16] R. C. R.BUYYA AND X.LI, *Autonomic cloud computing: Open challenges and architectural elements*, 3rd International Conference on Emerging Applications of Information Technology, Kolkata, India, (2012), pp. 3–10.

[17] S. R.BUYYA AND M.PALANISWAMI, *Internet of things (iot): A vision, architectural elements and future directions*, Future Generation Computer Systems, 29 (2013), pp. 1645–1660.

[18] I. STOJMENOVIC, *Fog computing: A cloud to the ground support for smart things and machine-to-machine networks*, in 2014 Australasian telecommunication networks and applications conference (ATNAC), IEEE, 2014, pp. 117–122.

[19] A. SUNYAEV AND A. SUNYAEV, *Internet computing*, Springer, 2020.

[20] H. ZHANG, Y. ZHANG, Y. GU, D. NIYATO, AND Z. HAN, *A hierarchical game framework for resource management in fog computing*, IEEE Communications Magazine, 55 (2017), pp. 52–57.