Optimizing Transfer Efficiency in Multi-Cloud Storage Systems with Edge and Fog Computing

Nitesh Bharot, Nisha Ghangare, Priyanka Verma
Data Science Institute, University of Galway, Ireland {firstname.lastname}@universityofgalway.ie

2023 IEEE 2nd Industrial Electronics Society Annual On-Line Conference (ONCON) | 979-8-3503-5797-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/ONCON60463.2023.10431110

***Abstract*—Multi-cloud environments have emerged as a promising solution for computing, offering scalability, fault tolerance, and vendor flexibility. However, efficient data transfer between multiple cloud providers and data owners is crucial for ensuring optimal performance and timely execution of workloads. This paper explores the challenges associated with data transfer rates in multi-cloud storage systems and discusses various optimization strategies to improve data transfer efficiency. It highlights the limitations of single-cloud storage systems and the unresolved issues related to data transfer in conventional models. To address these challenges, a comprehensive architecture is proposed, integrating multiple cloud storage services and leveraging fog and edge computing paradigms. The study aims to optimize the data transfer efficiency in multi-cloud storage systems.**

***Index Terms*—Multi-cloud, Transfer rate, Bandwidth, Fog computing, Edge computing**

I. Introduction

Cloud computing's inception has unlocked vast opportunities for the development and global provision of services, democratizing the digital space for individuals and small businesses [1], [2]. Despite these advancements, the reliance on single-cloud storage systems introduces substantial limitations, including capacity, bandwidth, geographical reach, and potential service disruptions. Furthermore, vendor lock-in confines users to the capabilities of a single provider, impacting the scalability and versatility of services. This reality underscores the insufficiency of conventional single-cloud models to cater to escalating storage demands in today's digital landscape. While cloud computing has been extensively researched, many unresolved issues persist [3], [4]. These include challenges related to proprietary APIs, the need for application architectures that effectively align with the underlying cloud environment [5], [6], and efficient data transfer. Hence, it's paramount to address these challenges and explore innovative strategies that optimize storage and data transfer efficiency in cloud environments.

There is no question that the research landscape has been completely transformed by the fast progress of cloud computing technology, which has made it possible for everyone to interact globally, access immense computational resources, and store and analyze massive information. But the idea of a single cloud provider could not be adequate to address the various needs of research groups as cloud computing develops. Thus, in order to provide improved scalability, fault tolerance, cost optimization, and data sovereignty, a multi-cloud system that incorporates different cloud providers is presented in this article as a comprehensive architecture. Our research aims to address the critical issues by exploring the potential of multi-cloud storage system in conjunction with edge and fog computing paradigms.

In 2010, Vukolic et al. [7] first suggested the idea of multi-cloud. Following that, a lot of work was done on the multi-cloud storage and database systems. DEPSKY [8] offers a methodology that divides files into many sections before sending them to the servers of various cloud companies. DEPSKY first uses its technique in a commercial cloud to prevent frequent service interruptions. The owner of the data is no longer to be concerned about vendor lock-in. Even if one cloud server fails, the owner of the data can still restore it all using other clouds. NCCloud [9] splits the file using regeneration codes, which require less repair traffic and retain the same fault tolerance and data redundancy as conventional erasure codes (like RAID-6) but with less repair traffic.

The prominence of information storage and retrieval in multi-cloud settings is on the rise. Sohal et al. [10] introduce a multi-cloud structure aimed at safeguarding user data. The main objective is to protect user data from potentially untrustworthy Cloud Service Providers (CSPs) who might share user data with malicious entities for personal gain. To fulfill this goal, data is segmented, encrypted, and dispersed across various clouds, employing client-side cryptography within their proposed structure.

A unified architecture was presented in [11] research to provide safe data exchange in a multi-cloud. Slice-based safe data sharing, similar to the concept above, was developed by Xu et al. [12] to enable secure data sharing in a multi-cloud. Since the meta table is still unprotected and this model does not accept video files, harmful insider attacks can occur.

Despite much research in the domain of multi-cloud platforms, there have been very few in the domain of data transfer in a multi-cloud platform. There are certain factors that intensely influence the transfer rates: (i) The latency introduced by the network infrastructure connecting the cloud providers significantly affects data transfer rates. High latency can lead to delays and suboptimal performance, especially when transferring data over long distances or across geographically dispersed clouds. (ii) The available bandwidth between cloud providers determines the maximum data transfer rate achievable. Limited bandwidth can result in bottlenecks and prolonged transfer times, impacting research workflows. (iii) The choice of data transfer protocols, such as TCP, UDP, or specialized transfer protocols like GridFTP or Aspera, can impact transfer rates. Different protocols have varying overheads,

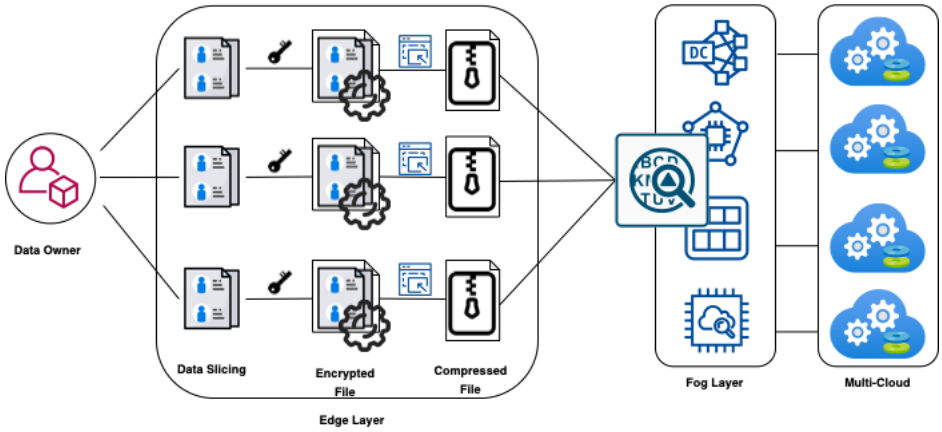

Fig. 1. Block diagram of the proposed framework

error recovery mechanisms, and throughput capabilities.

Keeping these in mind, we propose optimizing transfer efficiency in multi-cloud storage system in association with edge and fog computing framework. The key contributions of this paper are:

- Proposed framework presents a novel architecture integrating multi-cloud, edge, and fog computing, resulting in enhanced data transfer rate across diverse and distributed computing environments.
- Proposed work utilizes data compression algorithms to compress the data before it is transferred between clouds. Compressed data requires less bandwidth and can be transmitted more quickly, resulting in a optimized transfer rate.
- To mitigate security risks and safeguard data against unauthorized access, we have implemented the ChaCha20 encryption technique in the proposed framework.

II. Proposed Methodology

Data transfer is one of the critical components in a multi-cloud architecture. A large data file takes a significant amount of time to transfer to a multi-cloud or any storage architecture which in turn reduces the system's latency and causes delays. This also reduces the system's efficiency and affects the overall performance. Hence to overcome these issues, the proposed framework, not only provides efficient data transfer but also provides security. It enables the use of edge layers and fog layers to store data securely and efficiently on multi-cloud architecture. Additionally, it uses a error correction and detection mechanism which checks the files for tampering and enhances the utility of the system. The overall framework is described in detail as follows:

A. Architectural Overview

Figure 1 describes the architecture of the proposed framework for multi-cloud storage service. It consists of various layers which have their own computation mechanisms to allow secure and efficient data transfer. These are described as data owners, edge layer, fog layer, and multi-cloud storage system. The data owner, as the name suggests, is the owner of the data. It is responsible for uploading the data to multi-cloud platforms. A data owner could be a person or an organization and is believed to have a large amount of data to store on the cloud platform. Edge Layer is the first computation layer between the multi-cloud and the data owner. It is responsible for the initial computations of the data file at the user end. It is believed to be equipped with technologies like a data slicer, data encrypter, and data compressor. The fog Layer is the second computation layer between the multi-cloud and data owner. It is larger than the edge layer and is supposed to handle the major computations for a file before/after it is uploaded/downloaded to the multi-cloud architecture. It is believed to be equipped with error detection and correction mechanisms, data decompressors, data uploaders, and registries. Multi-cloud storage is a multi-story data storage system from different providers that is used to store large data files. It's an environment of multiple cloud storage services that work hand in hand to provide sufficient data storage abilities to the data owners. The number of cloud service providers depends upon the data owner and their overall cost to buy the resources.

B. Proposed Framework

The proposed framework is a secure data transfer methodology that uses edge computing and fog computing to enhance data transfer. Initially, the data owner chooses the number of cloud storage to store its files. Next, it uses the edge computing layer attached to it for data uploading. The edge layer first divides the data into multiple segments based on the number of cloud storage chosen. The data encryption mechanism present in the cloud next encrypts the file to detain any unauthenticated personnel to gain any information from the data. Then the data compression mechanism compresses the encrypted data to allow an efficient data transfer. Furthermore, the compressed data files are sent to the fog layer parallelly. The fog layer, which is connected just prior to the multi-cloud system receives the compressed data and decompresses it. It then uses the error detection and correction mechanism to

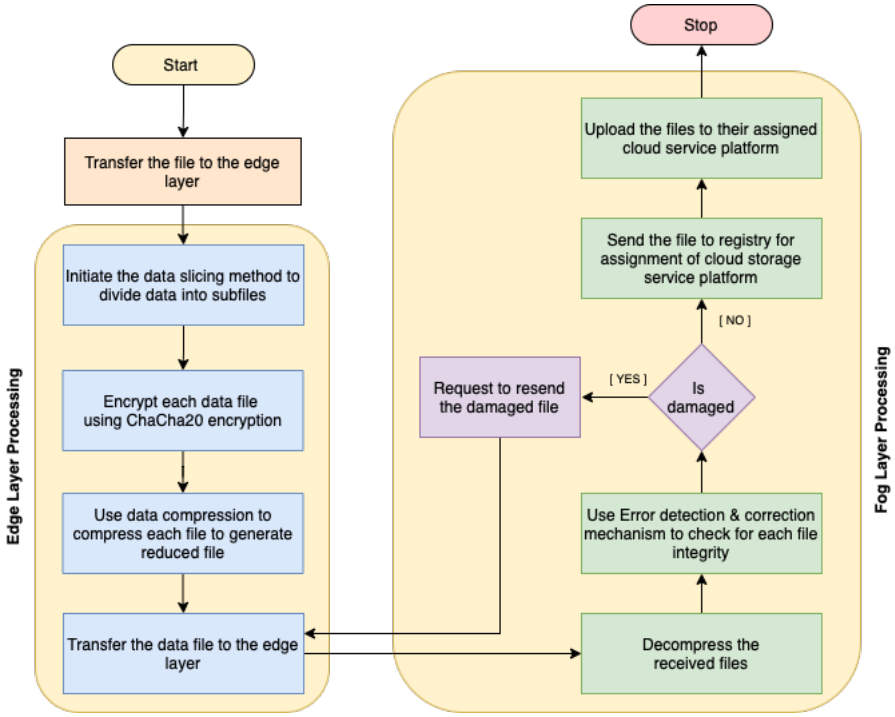

Fig. 2. Workflow of the proposed framework

check for each data file subpart consistency. If any of the sub-parts is tempered or received with error then it sends a request for re-transfer of the file. After affirmation, the correct encrypted file is passed through the registry where it gets assigned to a suitable cloud service platform using the scheduling algorithm. Finally, the data uploader uploads the files to desired cloud service. Figure 2 describes the workflow of the system. It enables efficient transfer with the help of multiple tools and functionalities such as data slicing, data encryption, data compression, data uploader, data decompressor, error correction-detection mechanism, and registry.

- Data slicing: It is the initial computation performed by the edge layer which divides the large data file into multiple files based on the number of cloud service platforms selected by the data owner. Data slicing can be advantageous when uploading data to a multi-cloud platform as it enables data distribution, optimizes bandwidth utilization, enhances fault tolerance, ensures compliance with data sovereignty requirements, and allows for cost optimization. By strategically dividing and distributing data subsets across multiple cloud providers, organizations can leverage the benefits offered by different clouds and create a robust and efficient multi-cloud infrastructure.
- Data encryption: It is the next computation required to enable data security. After data slicing an appropriate encryption technique is applied to the dataset. We used ChaCha20 [1] [13] to secure the data by encrypting it with a secret key. Data encryption plays a crucial role in protecting the confidentiality and integrity of data. It ensures that sensitive information remains confidential by encoding files with encryption algorithms, making them unreadable to unauthorized individuals. It provides an additional layer of security for data at rest, preventing unauthorized access to stored files. During transmission, encryption safeguards data from interception and eavesdropping. Compliance with data protection regulations is facilitated through encryption. Secure collaboration is enabled by sharing encrypted files, ensuring only authorized parties can access the content.
- Data compression: After encrypting the data, we used the zlib library of Python to implement the data compression. This enables us to reduce the size of the file to be transferred across the network. In many facets of data management and transmission, data compression is essential. By more effectively encoding data, it tries to minimize the size of data files or streams. Reduced storage needs, quicker data transfer rates, and better bandwidth utilization are just a few advantages of compression. As compressed files require less storage space, it facilitates effective data backup and archiving. Data compression is also necessary for multimedia applications since it reduces the size of audio, picture, and video files without significantly compromising their quality. Overall, data compression improves resource utilization, transmission effectiveness, and data management across a variety of fields.
- Data uploader: It deals with data uploading. Its functionality is to upload the data to the multi-cloud storage.

[1]https://pycryptodome.readthedocs.io/en/latest/src/cipher/chacha20.html

A multi-cloud data uploader simplifies the process of uploading data to multiple cloud storage providers. It integrates with various platforms, allowing simultaneous uploads to different clouds. Optimized network utilization, reduces upload time, and enhances efficiency. Data integrity and security are prioritized, often including encryption and integrity checks. The uploader offers progress monitoring and error handling for reliable uploads.

- Data decompressor: This section deals with the data decompression part at the fog layer. It is a tool that reverses the compression process by restoring compressed data to its original form. It utilizes specific decompression algorithms to expand compressed files and streams, allowing users to regain access to uncompressed data. Data decompressors support various compression formats, enabling compatibility with a wide range of compressed files. They optimize resource usage by efficiently extracting and utilizing compressed data, preserving data integrity throughout the decompression process.
- Error correction and detection mechanism: Mechanisms for error correction and detection are essential for maintaining the dependability and integrity of data reached to the fog layer for sending to multi-cloud storage service. These procedures are intended to locate and fix mistakes that could happen when data is being sent or stored. In order to find flaws in data, error detection and error correction is used. These techniques play a crucial role in preventing data corruption, improving the correctness of data that is sent or stored, and maintaining the dependability of storage and communication systems. By detecting and correcting errors in real-time, these mechanisms ensure that data integrity is maintained during transmission. This reduces the need for retransmissions and minimizes the overall time and bandwidth required for successful data transfer.
- Registry: The role of a registry that stores cloud_id and file_id index pairs at the fog layer is to facilitate efficient retrieval and management of files stored in a multi-cloud storage. By storing these index pairs, the registry acts as a centralized repository that maps file identifiers (file_id) to the corresponding cloud storage locations (cloud_id). This allows users or applications to easily locate and access specific files by querying the registry instead of searching through multiple cloud providers individually. The registry enhances file search, retrieval, and metadata management by providing a unified interface, simplifying data governance, and optimizing the overall file management process in a multi-cloud storage.

III. Experimental Results & Evaluation

The proposed schema deals to increase the transfer rate of the data thereby enhancing the data transfer system and reducing the system of latency issues. We conducted a series of rigorous experiments to determine the best compression technique and the best methodology to tackle the present problem scenario. As mentioned in the previous section the concepts of data slicing and data compression are the determined factors of the proposed framework which enhances the transfer rate. These experiments have been conducted with the help of Python 3.0, 1650Ti GTX integrated Intel i5 9th generation processor. Moreover, Google Drive, DropBox, & Mega cloud services are used to implement the multi-cloud storage system.

This section compares the proposed framework with multi-cloud storage systems against the traditional single cloud storage systems. The proposed framework consists of the inclusion of all steps mentioned in proposed methodology section. Whereas in traditional single cloud storage systems no such techniques such as data slicing and data compression were employed. However, to accommodate the security aspects and give a fair comparison with proposed we had included the data encryption phase in single cloud storage systems as well.

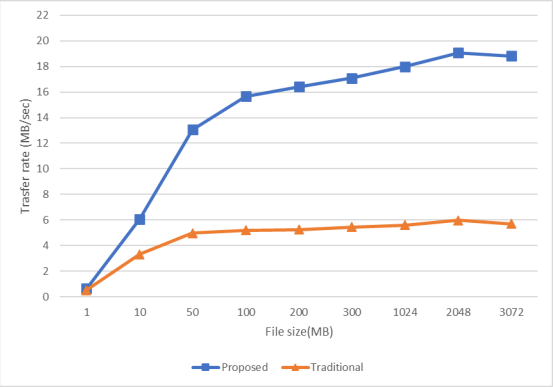

Fig. 3. Comparison of transfer rate of proposed and traditional system for different size CSV files

Figure 3 demonstrates the necessity of the proposed schema by comparing the transfer rate for a CSV file with different sizes. Transfer rate is described as the total time taken to transfer the file from edge to fog layer. It depicts that a considerable amount of increase in transfer rate could be achieved from 5.7 MB per second to 18.84 MB per second for 3GB csv data by implementing proposed framework. The observations from Figure 3 indicate that proposed framework aids to increase in transfer rate.

The proposed framework is also tested on text data for its transfer rate as shown in Fig. 4. It is observed that proposed framework act as a generalized mechanism for any data type to enhance the data transfer rate. This figure indicates an increase in the transfer rate from 6.07 MB per second to 18.13 MB per second for 3GB of text data.

Table I presents a comparison of different compression techniques in terms of time with the proposed framework. In the proposed framework, data compression is applied after data slicing and encryption. Thus, with the proposed framework compression time will be less in comparison to compression over the whole data file. The table consists of two file types, CSV and TEXT, and provides the file sizes for each type. The

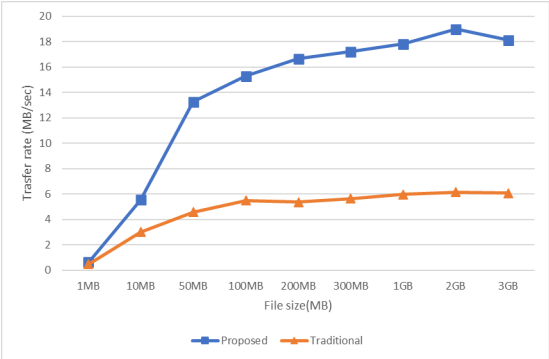

Fig. 4. Comparison of transfer rate of proposed and traditional system for different size TEXT files

TABLE I
COMPARISON OF COMPRESSION TECHNIQUES TIME (IN SECONDS) WITHIN PROPOSED FRAMEWORK

File type	File Size	ZLIB	GZIP	BZ2	LZMA
CSV	1MB	0.010	0.011	0.055	0.107
	10MB	0.089	0.098	0.461	0.876
	50MB	0.444	0.465	2.396	5.809
	100MB	0.153	0.172	0.270	0.448
	200MB	1.780	1.845	0.808	26.013
	300MB	2.628	2.806	13.166	38.169
	1GB	1.623	1.766	2.470	4.373
	2GB	16.742	18.270	83.088	243.191
	3GB	25.117	27.191	123.467	378.369
TEXT	1MB	0.014	0.119	0.048	0.082
	10MB	0.095	0.097	0.477	0.873
	50MB	0.448	0.457	2.271	6.993
	100MB	0.157	0.160	0.289	0.466
	200MB	1.776	2.150	9.143	27.247
	300MB	2.740	2.807	13.274	39.829
	1GB	9.197	9.726	44.445	134.908
	2GB	17.505	18.561	82.965	245.537
	3GB	25.967	28.172	122.013	365.578

compression techniques compared are ZLIB, GZIP, BZ2, and LZMA. The time measurements are given in seconds.

For the CSV file type, the following observations can be made:

- As the file size increases, the time taken by each compression technique is also increased. For example, for a file size of 10MB, ZLIB took 0.089 seconds, GZIP took 0.098 seconds, BZ2 took 0.461 seconds, and LZMA took 0.876 seconds.
- Among the four compression techniques, LZMA consistently took the longest time, especially for larger file sizes and ZLIB takes the least time.

For the TEXT file type, similar observations can be made:

- The duration required for each compression technique exhibited a direct relationship with file size. For instance, a 10MB file necessitated 0.095 seconds for ZLIB, 0.097 seconds for GZIP, 0.477 seconds for BZ2, and 0.873 seconds for LZMA.
- Consistently, LZMA required the most time for data slicing, particularly for larger files, while ZLIB proved to be the most time-efficient technique.

These results indicate that the LZMA compression technique tends to have the highest time requirements for data compression, while ZLIB and GZIP are generally faster. BZ2 falls in between the other techniques in terms of time taken. For larger files, ZLIB seems to take less time for compression. Thus, in the proposed framework we implemented ZLIB for data compression.

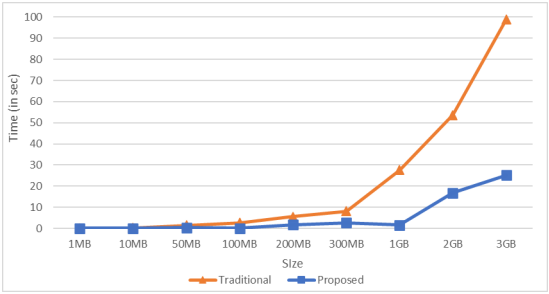

Fig. 5. Comparison of compression time with and without proposed framework

Figure 5 indicates that with proposed framework takes less time for the compression in comparison to the whole data. It establishes the fact that data compression time could be significantly reduced by applying the concept of parallelization.

TABLE II
COMPARISON OF ENCRYPTION TIME OF PROPOSED WITH TRADITIONAL FRAMEWORK

File type	File size	Proposed	Traditional
CSV	1MB	0.003	0.078
	10MB	0.010	1.983
	50MB	0.060	5.336
	100MB	0.093	6.187
	200MB	0.195	11.432
	300MB	0.315	18.857
	1GB	0.986	58.323
	2GB	2.023	76.743
	3GB	3.278	127.404
TEXT	1MB	0.004	1.434
	10MB	0.020	1.487
	50MB	0.122	2.652
	100MB	0.254	4.034
	200MB	0.524	7.153
	300MB	0.832	10.012
	1GB	2.715	31.574
	2GB	5.115	106.564
	3GB	9.649	259.120

Table II compares the encryption time for different file types with and without a proposed framework. The table includes two file types, CSV and TEXT, and provides the file sizes for each type. The encryption times are measured in seconds with an encryption technique as ChaCha20.

For the CSV file type, the following observations can be made:

- For a file size of 1MB, the proposed framework took 0.003 seconds, while the traditional approach (without proposed) took 0.078 seconds for encryption.

- As the file size increased, both traditional and proposed framework, the encryption time also increased. For example, for a file size of 10MB, proposed framework took 0.010 seconds, while traditional took 1.983 seconds.
- The proposed framework consistently resulted in faster encryption times compared to encryption time without proposed framework for traditional approach.

For the TEXT file type, similar observations can be made:

- The proposed framework significantly reduced encryption time for a 1MB file to 0.004 seconds, compared to the 1.434 seconds with traditional approach.
- As the file size expanded, encryption times escalated both with and without proposed framework. For instance, a 10MB file required 0.020 seconds with proposed framework and 1.487 seconds without.
- Consistently, the proposed framework demonstrated a quicker encryption process compared to the framework not utilizing the proposed method in traditional approach.

It is evident from the results that the TEXT and CSV files of the same sizes almost take similar time in both with and without proposed framework. However, a small difference in time could be seen between TEXT and CSV files of specific size due to the text complexity, punctuation, white spaces, and special characters of TEXT files.

IV. Conclusion & Future Work

The proposed framework presents an effective solution for enhancing data transfer efficiency in multi-cloud ecosystems. The empirical results corroborate that employing proposed framework within a multi-cloud storage service optimizes data transfer efficiency. This framework further leverages edge and fog computing paradigms, utilizing parallelization, data compression, data slicing, and encryption strategies to amplify overall system efficiency. These techniques substantially alleviate the burden on the data owner's end, enabling more streamlined operations and facilitating swifter data transfer. Moreover, proposed framework effectively addresses latency issues by offloading computational overhead to the edge and fog computing layers, thereby reducing network congestion and improving data flow. The incorporation of parallelization strategies distinctly augments data transfer speed, reinforcing the potential of proposed framework as a practical and efficient solution for enhancing data transfer efficiency in the evolving landscape of multi-cloud storage systems. The cumulative impact of these strategies not only streamlines the data transfer process but also underpins a more robust, secure, and efficient multi-cloud system, paving the way for future developments in cloud computing and data management. In the future, we would like to improve the compression abilities by also considering the concept of compression ratio in the framework.

Acknowledgment

The research leading to these results has received funding from the European Union's Horizon Europe research and innovation programme under Grant Agreement No. 101100680 (GN5-1), and also by grants from Science Foundation Ireland under Grant Numbers 16/RC/3918 and 12/RC/2289_P2. For the purpose of Open Access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission.

References

[1] S. Vinoth, H. L. Vemula, B. Haralayya, P. Mamgain, M. F. Hasan, and M. Naved, "Application of cloud computing in banking and e-commerce and related security threats," *Materials Today: Proceedings*, vol. 51, pp. 2172–2175, 2022.

[2] J. Weinman, *Cloudonomics+ Website: The Business Value of Cloud Computing*. Wiley Online Library, 2023.

[3] S. Bharany, K. Kaur, S. Badotra, S. Rani, Kavita, M. Wozniak, J. Shafi, and M. F. Ijaz, "Efficient middleware for the portability of paas services consuming applications among heterogeneous clouds," *Sensors*, vol. 22, no. 13, p. 5013, 2022.

[4] P. Verma, S. Tapaswi, and W. W. Godfrey, "A request aware module using cs-idr to reduce vm level collateral damages caused by ddos attack in cloud environment," *Cluster Computing*, pp. 1–17, 2021.

[5] M. Ciavotta, G. P. Gibilisco, D. Ardagna, E. Di Nitto, M. Lattuada, and M. A. A. da Silva, "Architectural design of cloud applications: A performance-aware cost minimization approach," *IEEE Transactions on Cloud Computing*, vol. 10, no. 3, pp. 1571–1591, 2020.

[6] D. L. Frink and B. K. Clore, "Customized memory modules in multi-tenant provider systems," Nov. 22 2022. US Patent 11,509,711.

[7] M. Vukolić, "The byzantine empire in the intercloud," *ACM Sigact News*, vol. 41, no. 3, pp. 105–111, 2010.

[8] A. Bessani, M. Correia, B. Quaresma, F. André, and P. Sousa, "Depsky: dependable and secure storage in a cloud-of-clouds," *Acm transactions on storage (tos)*, vol. 9, no. 4, pp. 1–33, 2013.

[9] H. C. Chen, Y. Hu, P. P. Lee, and Y. Tang, "Nccloud: A network-coding-based storage system in a cloud-of-clouds," *IEEE Transactions on computers*, vol. 63, no. 1, pp. 31–44, 2013.

[10] M. Sohal, S. Bharany, S. Sharma, M. S. Maashi, and M. Aljebreen, "A hybrid multi-cloud framework using the ibbe key management system for securing data storage," *Sustainability*, vol. 14, no. 20, p. 13561, 2022.

[11] D. Ardagna, E. Di Nitto, P. Mohagheghi, S. Mosser, C. Ballagny, F. D'Andria, G. Casale, P. Matthews, C.-S. Nechifor, D. Petcu, *et al.*, "Modaclouds: A model-driven approach for the design and execution of applications on multiple clouds," in *2012 4th International Workshop on Modeling in Software Engineering (MISE)*, pp. 50–56, IEEE, 2012.

[12] P. Xu, X. Liu, Z. Sheng, X. Shan, and K. Shuang, "Ssds-mc: slice-based secure data storage in multi-cloud environment," in *2015 11th International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness (QSHINE)*, pp. 304–309, IEEE, 2015.

[13] Y. Iqbal, M. F. Amjad, F. Khan, and H. Abbas, "The implementation of encryption algorithms in mqtt protocol for iot constrained devices," in *2022 14th International Conference on Computational Intelligence and Communication Networks (CICN)*, pp. 804–810, 2022.