This is a postprint version of the following published document:

D. Carrizales-Espinoza, D. D. Sánchez-Gallegos, J. L. Gonzalez-Compean and J. Carretero, "A Federated Content Distribution System to Build Health Data Synchronization Services," *2021 29th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP),* 2021, pp. 1-8.

DOI: 10.1109/PDP52278.2021.00011

©2021 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

A Federated Content Distribution System to Build Health Data Synchronization Services

Diana Carrizales-Espinoza
Cinvestav Tamaulipas
Victoria, Mexico
diana.carrizales@cinvestav.mx

Dante D. Sánchez-Gallegos
Cinvestav Tamaulipas
Victoria, Mexico
Universidad Carlos III de Madrid
Madrid, Spain
100433984@alumnos.uc3m.es

J. L. Gonzalez-Compean
Cinvestav Tamaulipas
Victoria, Mexico
joseluis.gonzalez@tamps.cinvestav.mx

Jesus Carretero
Universidad Carlos III de Madrid
Madrid, Spain
jcarrete@inf.uc3m.es

***Abstract*—In organizational environments, such as in hospitals, data have to be processed, preserved, and shared with other organizations in a cost-efficient manner. Moreover, organizations have to accomplish different mandatory non-functional requirements imposed by the laws, protocols, and norms of each country. In this context, this paper presents a Federated Content Distribution System to build infrastructure-agnostic health data synchronization services. In this federation, each hospital manages local and federated services based on a pub/sub model. The local services manage users and contents (i.e., medical imagery) inside the hospital, whereas federated services allow the cooperation of different hospitals sharing resources and data. Data preparation schemes were implemented to add non-functional requirements to data. Moreover, data published in the content distribution system are automatically synchronized to all users subscribed to the catalog where the content was published.**

***Index Terms*—Data synchronization, federated content distribution, health data, cloud computing**

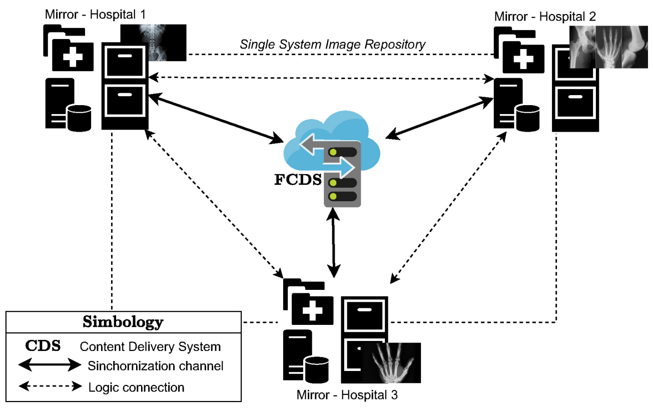

Fig. 1. Federated content distribution system

I. Introduction

Data volume produced and managed by organizations has been growing over the past years because the end-users associated with organizations produce, store, and use data consistently and continuously, which produces a data accumulation effect [1]. In organizational environments, such as in hospitals, data have to be processed, preserved, and shared with other organizations in a cost-efficient manner. Also, organizations have to accomplish different mandatory non-functional requirements imposed by the laws, protocols, and norms of each country. These requirements include the security of the data during their transportation and preservation, establishing controls over the access of the data (privacy) and resources (control access), the integrity and confidentiality of the data, as well as their reliability.

In the past, we developed different solutions that accomplish the non-functional requirements required in use cases such as satellite imagery and geospatial data management. These solutions are HIDDRA [2], the Branch replication scheme (BRS) [3], SkyCDS [4], and FedIDS [5]. However, these solutions are static in terms of the non-functional requirements that they could manage, so the users have to perform manual modifications in the code of the solutions to add, delete, or modify the methods used to accomplish the requirements managed by the solution. This is important because, in public cloud-based solutions, users must face challenges such as security, vendor-lock in, outages, and integrity or confidentiality violations, which could not be present in private clouds or private clusters of computers inside of the organization.

In this paper we present a Federated Content Distribution System (*FCDS*) used to build health data synchronization services that are technology agnostic of the infrastructure, allowing organizations to deploy their services in any of private, public, or hybrid cloud. Figure 1 depicts a conceptual representation of the *FCDS*, which allows sharing data (i.e., tomographies, mammograms, and magnetic resonances) in a synchronous manner between three hospitals through a Content Delivery Network (CDN) [4], [6]. The motivational idea for the federation of services is that organizations can have governance over their services, data, infrastructure, and

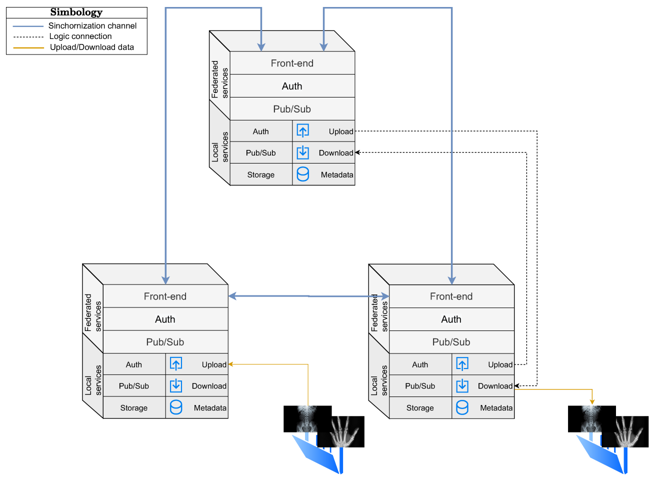

Fig. 2. Detailed system architecture

the non-functional requirements by including methods to accomplish them. Moreover, the members of the federation can uniquely create a set of services to allow the participants to share resources and data. The services include data preparation and recuperation schemes [7] to pre-process the data before distributing them to other participants in the federation or to the cloud, applying different filters that accomplish the non-functional requirements required by an organization. The main advantages of the proposed solution are high-reliability, load distribution, data integrity, data confidentiality, and vendor independence.

The structure of the paper is as follows: in Section II we present the system architecture. Section III shows evaluation results of the solution proposed. Section IV includes related work. Section V includes main conclusions of the paper.

II. A Federated Content Distribution System

In this section, we describe the design principles of our federated content distribution system (FCDS). The architecture and main components of the system are presented, as well as schemes based on containerized parallel patterns to prepare/retrieve the data when it is upload/download through the FCDS. These schemes manage the non-functional requirements required by organizations to accomplish regulations imposed by organizations and governments.

A. System architecture

Figure 2 shows the detailed architecture and main components of the *FCDS* for three different hospitals, so as the stacked services of three different organizations.

Each organization (hospital) deploys two types of services: federated services and local services. The purpose of distinguishing these two types of services is that each organization has governance over its data and resources, but, at the same time, some services allow organizations to share resources and data transparently and securely. Federated services are all those that are visible to other organizations. Through these services, organizations have access to the resources and data, to which they have authorized access, available in other organizations (medical images or medical records). For this, the federated services for each organization are, at least, the following:

- **Front-end**: The layer through which clients and participants of the federation access the services of the organization. It has an API that allows interaction with other participants, as well as token validation and a proxy to redirect requests the services in the lower levels or available in other organizations in the federation.
- **Auth**: This layer is responsible for validating the access of external users, that are members of the federation, to a specific organization.
- **Pub/Sub**: It is responsible for managing the requests for publication and subscription to data (images or medical records) from users belonging to the federation.

Local services are responsible for managing the system within each organization, including internal user management, as well as the data handled in the organization. The idea of those services is that each organization can manage its data catalogs independently of those published in the federation. In the event that users want to publish their data in the federation, they can do it by only changing the status of the catalog from "*local/private*" to "*federated*", or similarly by giving access only to a certain group of users in the federation. Within the local services, three layers are included for managing the authentication of users and tokens of the organization, managing publications and subscriptions, and managing storage, as well as the metadata associated with these services. Moreover, these services include two clients: one for loading and one for downloading data. The loading client is responsible for preparing the data for publication and subsequently sending them through the content delivery network to all those users who have subscribed to the data, both within the organization and in the federation. Meanwhile, the downloading client is responsible for performing the reverse process: to obtain the data from the *CDN* and to retrieve them so that users can view and manipulate them. The idea of these clients is that they allow efficient and secure synchronization of data between organizations automatically.

The exchange of messages and metadata between the federation members is done through HTML Rest APIs, whereas the data is transported by using a content delivery network called SkyCDS [4]. Data preparation schemes [7] are deployed to add non-functional requirements to the data before being transported. These schemes process the data to add properties such as cost-efficiency, reliability, and security before sharing the data with other organizations. Similarly, data retrieval schemes are implemented to retrieve the original data by applying the inverse operations of the preparation schemes (for example, decompressing the data compressed in the preparation scheme).

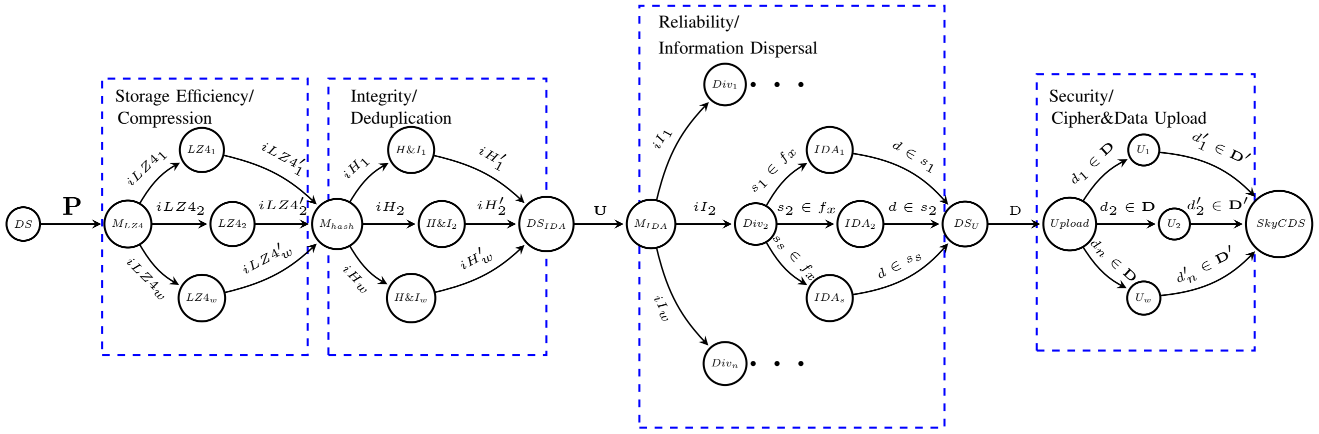

Fig. 3. DAG for data preparation scheme.

B. Non-functional requirements fulfillment by using data preparation schemes

In real-scenarios of data management, non-functional requirements (i.e., security, efficiency, and reliability) must be fulfilled according to different health management norms (i.e., NOM-024-SSA3-2010 and NOM-004-SSA3-2012) and laws imposed by governments and organizations [8], [9]. In this context, in this section, we present a data preparation scheme to add non-functional properties to the data before being uploaded for storage or transmitted by using non-controlled environments such as the Internet and the cloud [10]–[12].

The non-functional requirements added with these data preparation/retrieval schemes are cost-efficiency, reliability, and security.

- Cost-efficiency is achieved by using compression and deduplication techniques to reduce the number of contents or amount of data to be pre-processed. Also, these techniques reduce the data sent to the cloud and the costs of the outsource data management tasks.
- Reliability is added to mitigate problems caused by outages in the infrastructure where the data are processed and stored [13], [14]. This requirement is achieved by using the well-known IDA algorithm [15].
- Security (CIA: confidentiality, integrity, and access control) is added to solve problems that arise when the data is shared through non-controlled and untrusted environments, such as in the cloud [16], [17].

To add these non-functional requirements, data are prepared through a set of stages creating dataflows in pipelines. Figure 3 shows a directed acyclic graph (DAG) of the stages considered in the preparation schemes. In the DAG, bifurcations producing parallelism, are created when parallel patterns are used in a given stage. See the $Manager/Worker$ (M_{hash}) at the deduplication stage and $D\&C$ (Div_n) at the reliability stage in the DAG of the data preparation scheme.

To provide cost-efficiency, compressing, and deduplication stages were implemented. The compressing stage is based on the LZ4 algorithm. The deduplication service for storage uses a hash algorithm h (SHA-3-256 [18]) to generate a fingerprint $h(c)$ of the content and to detect the duplicate data as follows: $if(C\_1 = C\_2)$ then $h(C\_1) = h(C\_2)$. In this service, each hash is calculated and compared to the hashes previously discovered, which allows identifying replicated content before sending them to the data preparation scheme. Also, a set of hashes constitute the metadata. New hashes (unique data) are indexed in a database implemented in MongoDB, and the content is sent to the data preparation scheme. When the service detects a duplicate hash, it rejects the content because these hashes already have been pre-processed in the past.

To provide the reliability property to the preparation/retrieval scheme, the information dispersal algorithm (IDA) [15] was used. In the preparation scheme, each file is segmented into n segments, which are called dispersal files, with redundancy. These segments are stored into n storage locations and the retrieval scheme recovers the original files by downloading m of the segments, where $n < m$.

Data arriving at the security stage are sent to an encrypt and forwarding daemon of SkyCDS [4]. This daemon executes ciphering techniques based on AES [19] and CP-ABE [20] to encrypt the incoming data and to create a ciphered object including access controls. This is established over the data before sending them to the cloud.

C. Efficiency based on containerized parallel patterns

The data preparation stages are executed by using parallel patterns based on virtual containers to improve the efficiency of the data preparation process. In Figure 3 these patterns can be observed in the form of bifurcations creating two different patterns: manager/worker and divide&conquer.

1) Manager/Worker pattern: The Manager/Worker pattern processes data in different phases such as cloning, task definition/distribution, and task execution supervision. In the cloning phase, the $manager$ creates *VCs* instances, each one corresponding to a given processing stage included in either the preparation or retrieval schemes. Those clones are called $workers$.

In the task definition/distribution phase the $manager$ reads the content stored in a data source direction (DS). For this, the $manager$ creates a list, by using a set of paths ($P = \{Path_1, Path_2, \ldots, Path_n\}$), and each path is distributed in a load-balanced manner to the workers by using the two choices algorithm, which allows choosing the worker with low workload adding the efficiency property to task [21].

In the supervision phase, the $manager$ verifies that the $workers$ deliver the results of the task assigned to them to the next stage in the schemes. Each $worker$ performs the hashing and indexing stages over a set of contents allocated to it by the $manager$. The $workers$ calculate the hash of contents and the hashes are indexed in a metadata service. The indexing service stores the unique hashes in a table to keep the consistency of files previously indexed, which allows adding the integrity property to the data.

2) Divide&Conquer: The Divide and Conquer pattern is composed of two main components: i) $divide$ and ii) $conquer$. $Divide$ is a software instance that splits the data into s segments, which are processed by s workers that are clones that execute the very same processing task. Whereas the $conquer$ consolidate the results of each $worker$ into a single result to deliver them to the next processing stage or the cloud.

D. Prototype implementation details

A **data preparation engine** (DPE) component is used to deploy the stages required to add non-functional requirements. In this engine, all components of the preparation/retrieval scheme (including deduplication system, indexing system, and pre-processing applications) are encapsulated into a set of virtual containers (*VCs*). The *VCs* add portability property to the preparation/retrieval scheme and these allow encapsulate all software components of the preparation/retrieval schemes with their dependencies (i.e., libraries, packages, OS, environment variables), which enables the schemes to be deployed on different platforms. Moreover, the *VCs* are managed in the form of Building Blocks (*BBs*), where a Building Block is a software piece that provides a service, algorithm o function of the system. The *BBs* can be coupled by using I/O interfaces (i.e., memory, network, file system) to create a DAG [22], as shown in Figure 3.

This scheme was deployed in a prototype where the components such as the deduplication system, the IDA component, and the hash algorithm h (SHA-3-256 [18]) were implemented mainly in C language. The clients implementing the data preparation and retrieval schemes were also developed in C. The integrity verification metadata service was developed as a web service and uses a Mongo database deployed in the cloud [23]. The building blocks and the parallel patterns were created by using a framework based on Kulla [24]. The content delivery was implemented as a client and multi-cloud storage service instances provided by SkyCDS [4] (which is developed by using web services). An LZ4 [25] compression utility, programmed in C, and a cryptosystem based on AES [19] and CP-ABE [20], that were programmed in Java, were added to SkyCDS catalogs. All the components of the preparation and reliability schemes were encapsulated into *VCs* created by using the Docker platform.

TABLE I
CHARACTERISTICS OF THE DATASETS EVALUATED.

Dataset name	Image type	Size (GB)	No. files	Avg. file size (MB)
QIN-Breast	MR, CT	11	102451	14.5
CPTAC-PDA	CT, MR, DX, CR, US, PA	30	2070	14.5
DDSM	CT	47	1865	25.5

TABLE II
INFRASTRUCTURE USED FOR EXPERIMENTATION.

Compute	Center	RAM (GB)	Cores	Storage
Compute 1	Hospital 1	64	12	5.4 TB
Compute 2	Hospital 2	64	12	2.7 TB
Compute 3	Hospital 3	12	6	256 GB
EC2-1	Cloud node	32	16	600 GB

III. EVALUATION

This section presents a detailed description of the methodology applied to conduct the experimental evaluation of the solution proposed in this paper. This methodology is composed of three phases, where the flexibility and feasibility of the proposed system are evaluated. We set up three experimental scenarios including medical datasets to measure key performance indicators of each component of the content delivery and retrieval system.

A. Test datasets

Table I shows the main features of the medical images' datasets. The datasets include medical images such as mammography (CT), magnetic resonances (MR), clinical diagnosis (DX), computed radiographies (CR), ultrasounds (US), and pathologies (PA). The QIN-Breast [26] and CPTAC-PDA [27] datasets were obtained from "The Cancer Imaging Archive". The datasets are composed of a set of common disease images such as patient outcomes, treatment details, genomics, and image analyses. The file format used for the images is DICOM (Digital Imaging and Communications in Medicine) [28]. The Digital Database for Screening Mammography (DDSM) [29] dataset contains a set of mammography imagery, stored in PNG format.

B. Test Environment

The experimental environment was build using three remote centers connected through the Internet (see Figure 4), which are represented as three different machines (see Table II) and a content delivery network deployed in a cloud, which is in charge of transferring the contents through the different machines.

The experiments to validate our solution were conducted in three phases: *i)* In the first phase, the repositories were processed by using the preparation schemes varying the number of workers in the parallelism patterns; *ii)* In the second phase, we evaluated the download of data from the CDN and their recuperation by using the retrieval scheme; *iii)* In the third phase, the performance of the schemes are compared with solutions of the state-of-the-art. *Rsync* Rsync, a traditional solution for sending data to the cloud in a syncing manner during the sharing of data through different infrastructures. This solution does not consider the pre-processing of data, the establishment of attribute-based access controls, the verification of integrity, or the establishment of fault-tolerance. *SCP*, a solution that allows sending data to the cloud in a secure manner. This solution is based on a public/private key model, which includes the preparation of data during the data transport end-to-end and presents the restrictions of Rsync in terms of non-functional requirements. *TAR + SCP*, a solution used to compress and transport data between computers in a secure manner.

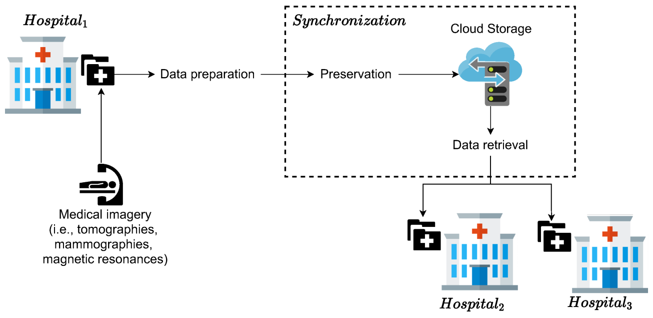

Fig. 4. Scenario used for experimentation.

To conduct these three phases of the experimental evaluation, the scenario depicted in Figure 4 was used. In this scenario, a hospital is producing medical imagery such as tomographies, mammographies, and magnetic resonances. These images are prepared and uploaded by using the FCDS, making them available for the other hospitals in the federation.

C. Experimental results

In this section, the results of the experiments performed in the three evaluation phases previously described are presented and analyzed.

1) Phase 1: Evaluating the preparation of data: In this phase we evaluated the performance of preparation schemes using the datasets allocated in different computers and sending the prepared data to the CDN, thus making data available for other members in the federation. We processed the different datasets in different centers to see the influence of the dataset and the computers associated. The major metric measured was the service time spent to process the dataset while varying the number of workers in the patterns. Figure 5 shows the service time when processing the QIN-Breast dataset stored in Compute3. As shown in Figure 5, a speedup of 4.7x is achieved with 8 workers to pre-process the 11 GB from the dataset. The execution time is larger than for the other datasets, due to a large number of small files.

Figure 6 corresponds to the service time spent by the preparation data scheme to process the CPTAC-PDA dataset stored in Compute1. In Compute1 the scheme spent 11.07 minutes for pre-processing 30 GB by launching only one worker. This produced a throughput of 45.16 MB/s, whereas

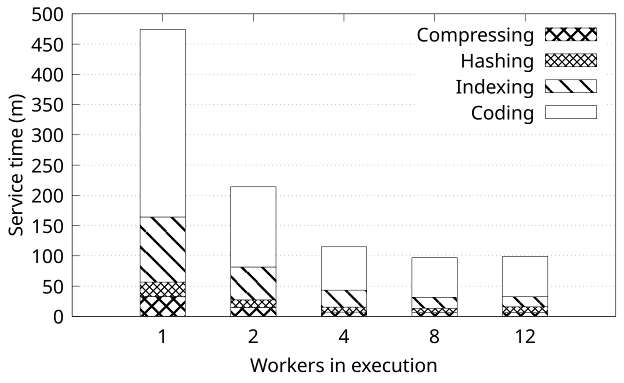

Fig. 5. Service time spent by the preparation scheme to process the QIN-Breast dataset.

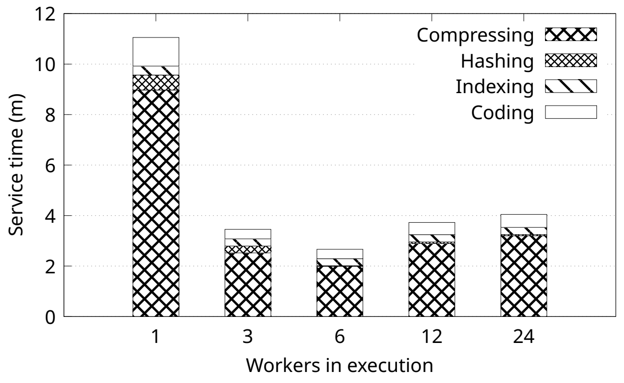

Fig. 6. Service time spent by the preparation scheme to process the CPTAC-PDA dataset.

the Master/Worker with 6 workers completes the same task in 2.76 minutes. This means an improvement of 75% with a speedup of 4x.

In a similar fashion, Figure 7 shows the service time to process the DDMS dataset by the preparation schemes stored in Compute2. In this case, the scheme pre-processed 47 GB with an improvement of 82% with an acceleration of 5x with 12 workers in comparison with 1 worker. As expected, the parallelism implemented in the schemes reduced the service time of each pre-processing task.

Moreover, from the results, it can be observed that the service time starts to increase when the number of workers is larger than the number of physical cores. The time spent to upload the datasets to the cloud was of 72 minutes for the CPTAC-PDA dataset in Compute1, and 160 minutes for the DDSM data in Compute2. As may be seen, the critical factor for the cost of those solutions is, not only the dataset size but mainly the number of files in the dataset.

2) Phase 2: Evaluating the costs of data retrieval: The data retrieval process is automatically triggered when the upload of the data is completed, and the CDN is in charge of delivering the data to the medical institutions subscribed to download the contents uploaded. In this experiment, we evaluated the time

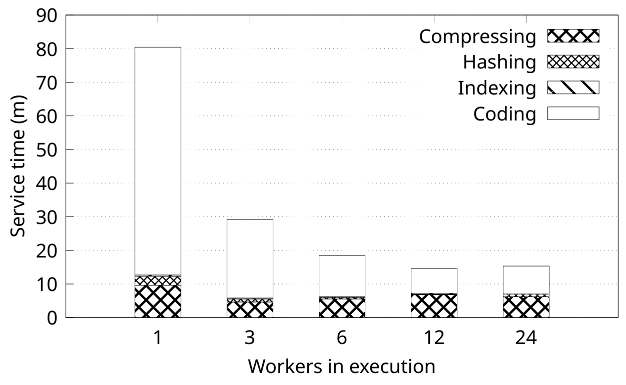

Fig. 7. Service time spent by the preparation scheme to process the DDMS dataset.

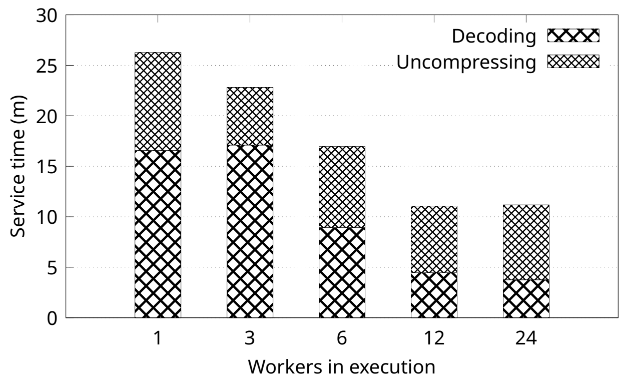

Fig. 8. Service time spent by the retrieval scheme to process the downloaded data.

to download the DDSM dataset, uploaded by Hospital 2 by using Compute2, to Hospital 1. This process was performed by using Compute1.

The time spent by the CDN client to download the data from the cloud storage was 64 minutes. After the data is downloaded by this client, the retrieval process of the original data starts, this is to decode the files from 4 of the 5 segments and decompress the files retrieved. Figure 8 depicts the results obtained from this retrieval process. As it can be observed in Figure 8, the service time is reduced by increasing the number of workers in the pattern. For example, the pattern with 12 workers spent 11.04 minutes to retrieve the data, whereas with only 1 worker spent 26.25 minutes, which means an improvement in the service time of 57.85%.

3) Phase 3: Performance comparison: Finally, we performed a direct comparison with three data transference applications from the state-of-the-art: SCP, TAR+SCP, and rsync. For this experiment, we used the dataset DDSM, transferring the data from Compute2 to Amazon EC-2.

Figure 9 shows the service time observed for each solution evaluated. In the case of SCP, we evaluated the transference by sending all the files without compressing the dataset, and in a second instance by compressing the dataset in a single

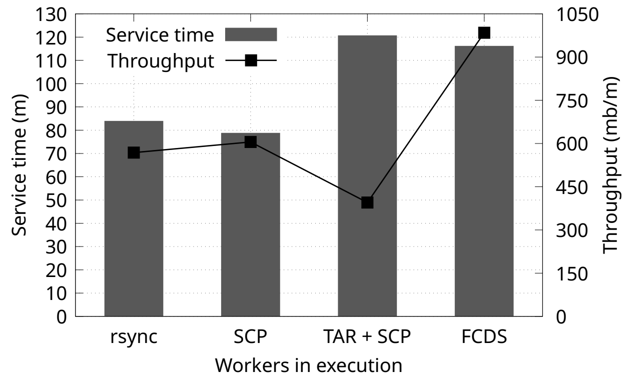

Fig. 9. Service time and throughput for upload the dataset DDSM by using different solutions from the state-of-the-art.

file. As can be seen in Figure 9, the solution proposed in this paper (*FCDS*) yields a higher throughput in comparison with the other three solutions, despite this solution yields the higher service time. The service time of our solution is of 116.10 minutes, whereas rsync spent 83.86, SCP 78.74, and TAR+SCP 120.62 minutes. The above is due to our solution loads a longer amount of data in comparison with the other solutions due to the redundancy added (1.7x). However, redundancy adds to the data the feature of reliability and fault-tolerant recovery.

IV. Related work

In previous works, we developed different solutions that solve different challenges, such as availability, storage usage, efficiency in download engine, and sharing information across different sites to mention some. These solutions were evaluated in use cases, such as satellite imagery and geospatial data management, and they are HIDDRA [2], BRS [3], SkyCDS [4], and FedIDS [5]. HIDDRA provides a highly efficient parallel multi-protocol download engine, using a *pub/sub* policy that helps the final user to obtain data of interest transparently. This solves the problem to distribute data from different institutions, such as NASA, ESA, or JAXA, to the scientific community aiming to fulfill the requirements of the final user [2]. FedIDS emerged as an improved version of HIDDRA. It is a federated cloud storage architecture that enables organizations to build shared data infrastructure through a federation of independent cloud resources to withstand outages, as well as avoid violations of integrity/confidentiality of images and data in sharing information, and collaboration workflows [5].

BRS is a model for data replication on a large scale. This model provides three main advantages over traditional approaches: increasing data access performance, optimizing storage usage, and providing the possibility to modify the replicas [3], [30]. To minimize the side effects of temporary, permanent cloud-based service outages and vendor lock-in, we developed SkyCDS, a resilient content delivery service based on a publish/subscribe overlay diversified cloud storage. SkyCDS lowers the overhead of the content dispersion and retrieving processes by taking advantage of multi-core technology [4].

Recently, Google has patented a federated system for centralized management and distribution of content media [31]. However, it differs from our solution in that it relies on a centralized node, while our solution is composed of peers. Moreover, in the literature, federated health data solutions have been proposed for processed, preserved, and sharing data in a confidential and cost-effective way across multiple sites.

Campi et al. [32] present a paper on securing communication among federated health information systems, which aims at describing how federated health information systems can offer security properties by adopting proper mechanisms to protect exchanged data and provided functionalities from malicious manipulations. Kumar et al. [22] proposed a federated health information architecture (HIA) that enabling health care providers and policymakers to use data for decision-making.

Nevertheless, these solutions are static in terms of the non-functional requirements added to the data before to be transported and stored. In this context, our solution (*FCDS*) enables organizations and end-users to choose in a flexible manner the non-functional requirements required by them, and to improve the processing of the data, parallel patterns based on virtual containers are used.

End-to-end solutions have been widely studied and developed to prepare data by adding non-functional requirements for mitigating risk just before sending them to different environments such as the cloud or the fog [19], [33]–[36], as data have to be moved between different infrastructures, to prevent confidentiality violation, content alteration and repudiation of events by omissions and unavailability of data or resources in environments (at any of the edge, fog or cloud).

End-to-end solutions can provide efficiency characteristic for data processing and data storage. Data processing techniques such as parallelism based on threads model, parallel patterns (e.g., manager/worker or divide&conquer), and in-memory schemes are used for improving the performance of data processing and reducing the impact of bottlenecks on pipelines or workflows. Examples of implicit parallelism based on threads, include workflows engines such as Makeflow [37], DagOn* [38], and Pegasus [39]. Whereas, examples of in-memory schemes for exchange data in an efficient manner are Sacbe [35] and Gearbox [40].

Data storage efficiency techniques are applied to reduce the volume of data to transport and store. In this context, data compression [41] and deduplication [42]–[44] are examples of techniques applied to this end.

Integrity solutions have also been proposed to verify the consistency of data in end-to-end connections, which allows detecting alterations to the data during its transport through different infrastructures [45]. Data integrity solutions consider applying checksums techniques (i.e., SHA2 hashing, SHA3 hashing, or MD5) to the data before its transmission to detect alterations [46]. Confidentiality services have been proposed to preserve data privacy when the data are transported and stored through non-controlled environments [47]. Whereas, access control techniques ensure that the data are only accessible by authorized users [19]. Examples of these services have been proposed by Wiseman et al. [41], Zhang et al. [48], and Morales et al. [19]. Data fault tolerance by using information dispersal and data codification techniques [35], [49] are examples of reliability solutions.

V. Conclusions

In this paper, we have presented a Federated Content Distribution System (*FCDS*) used to build health data synchronization services that can allow organizations to deploy their services in any private, public, or hybrid cloud. With *FCDS*, organizations can have governance over their services, data, infrastructure, and non-functional requirements, including methods to accomplish them. Resources and data can be shared through *FCDS* services, including aspects such as data preparation and recuperation schemes to pre-process the data before distributing them, applying filters that accomplish the non-functional requirements required by an organization (e.g., privacy). All those services can be composed of workflows of containerized applications. The main advantages of the proposed solution are high-reliability, load distribution, data integrity, data confidentiality, and vendor-independent.

The results of the experiments made show good scalability and good performance compared with traditional solutions used. Our experiments also show that the critical metric increasing the distribution time is the number of files in a dataset, more than the size of the files themselves. This should not be a problem in our environment, as the hospitals will not produce thousands of images every day. However, it may be in other environments. Thus, as future work, we will investigate how to group files using some ontology to work with subtrees in the ontology, those reducing the number of files to be exchanged. Traceability and security will be also explored to increase the performance of our solution.

Acknowledgment

This work has been partially supported by the grant “CABAHLA-CM: Convergencia Big data-Hpc: de Los sensores a las Aplicaciones” (Ref: S2018/TCS-4423) of Madrid Regional Government; the Spanish Ministry of Science and Innovation Project ” New Data Intensive Computing Methods for High-End and Edge Computing Platforms (DECIDE)”. Ref. PID2019-107858GB-I00; and by the project 41756 “Plataforma tecnológica para la gestión, aseguramiento, intercambio y preservación de grandes volúmenes de datos en salud y construcción de un repositorio nacional de servicios de análisis de datos de salud” by the FORDECYT-PRONACES.

References

[1] David Reinsel-John Gantz-John Rydning. The digitization of the world from edge to core. *Framingham: International Data Corporation*, 2018.

[2] Higuero, Tirado, Carretero, Félix, and de La Fuente. Hiddra: a highly independent data distribution and retrieval architecture for space observation missions. *Astrophysics and Space Science*, 321(3-4):169–175, 2009.

[3] Pérez, García-Carballeira, Carretero, Calderón, and Fernández. Branch replication scheme: A new model for data replication in large scale data grids. *Future Generation Computer Systems*, 26(1):12–20, 2010.

[4] Gonzalez, Perez, Sosa-Sosa, Sanchez, and Bergua. Skycds: A resilient content delivery service based on diversified cloud storage. *Simulation Modelling Practice and Theory*, 54:64–85, 2015.

[5] Gonzalez-Compean, Sosa-Sosa, Diaz-Perez, Carretero, and Marcelin-Jimenez. Fedids: a federated cloud storage architecture and satellite image delivery service for building dependable geospatial platforms. *International Journal of Digital Earth*, 11(7):730–751, 2018.

[6] AWS CloudFront. Amazon cloudfront. *URL: http://aws. amazon. com/cloudfront*, 2014. Accedido el 15 de julio, 2019.

[7] Carrizales, Sánchez-Gallegos, Reyes, Gonzalez-Compean, Morales-Sandoval, Carretero, and Galaviz-Mosqueda. A data preparation approach for cloud storage based on containerized parallel patterns. In *International Conference on Internet and Distributed Computing Systems*, pages 478–490. Springer, 2019.

[8] Hernández Mier and Torre Delgadillo. Regulación del acceso al expediente clínico con fines de investigación en méxico. *Revista CONAMED*, 22(1):27–31, 2018.

[9] Phillips. International data-sharing norms: from the oecd to the general data protection regulation (gdpr). *Human genetics*, 137(8):575–582, 2018.

[10] Cérin, Coti, Delort, Diaz, Gagnaire, Gaumer, Guillaume, Lous, Lubiarz, Raffaelli, et al. Downtime statistics of current cloud solutions. *International Working Group on Cloud Computing Resiliency, Tech. Rep*, 2013.

[11] Sloan and Warner. *Unauthorized access: The crisis in online privacy and security*. CRC press, 2013.

[12] Zissis and Lekkas. Addressing cloud computing security issues. *Future Generation computer systems*, 28(3):583–592, 2012.

[13] Gunawi, Hao, Suminto O, Laksono, Satria, Adityatama, and Eliazar. Why does the cloud stop computing?: Lessons from hundreds of service outages. In *Proceedings of the Seventh ACM Symposium on Cloud Computing*, pages 1–16. ACM, 2016.

[14] Bala and Chana. Fault tolerance-challenges, techniques and implementation in cloud computing. *International Journal of Computer Science Issues (IJCSI)*, 9(1):288, 2012.

[15] Ricardo Marcelín-Jiménez, Jorge Luis Ramírez-Ortíz, Enrique Rodríguez De La Colina, Michael Pascoe-Chalke, and José Luis González-Compeán. On the complexity and performance of the information dispersal algorithm. *IEEE Access*, 8:159284–159290, 2020.

[16] Bhushan and Gupta. Security challenges in cloud computing: state-of-art. *International Journal of Big Data Intelligence*, 4(2):81–107, 2017.

[17] French-Baidoo, Asamoah, and Oppong. Achieving confidentiality in electronic health records using cloud systems. *International Journal of Computer Network and Information Security*, 10(1):18, 2018.

[18] Morris J Dworkin. Sha-3 standard: Permutation-based hash and extendable-output functions. Technical Report 202, NIST Pub Series. Federal Inf. Process. Stds. (NIST FIPS), 2015.

[19] Morales, Gonzalez, Diaz, and Sosa. A pairing-based cryptographic approach for data security in the cloud. *IJISP*, 17(4):441–461, 2018.

[20] Odelu, Rao, Kumari, Khan, and Choo. Pairing-based cp-abe with constant-size ciphertexts and secret keys for cloud environment. *Computer Standards & Interfaces*, 54:3–9, 2017.

[21] Michael Mitzenmacher. The power of two choices in randomized load balancing. *IEEE Transactions on Parallel and Distributed Systems*, 12(10):1094–1104, 2001.

[22] Kumar, Mostafa, and Ramaswamy. Federated health information architecture: enabling healthcare providers and policymakers to use data for decision-making. *Health Information Management Journal*, 47(2):85–93, 2018.

[23] Reyes-Anastacio, Gonzalez-Compean, Morales-Sandoval, and Carretero. A data integrity verification service for cloud storage based on building blocks. In *2018 8th International Conference on Computer Science and Information Technology (CSIT)*, pages 201–206. IEEE, 2018.

[24] Reyes, Gonzalez, Morales, and Carretero. A data integrity verification service for cloud storage based on building blocks. In *2018 8th International Conference on Computer Science and Information Technology (CSIT)*, pages 201–206, July 2018.

[25] Bartík, Ubik, and Kubalik. Lz4 compression algorithm on fpga. In *2015 IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*, pages 179–182. IEEE, 2015.

[26] Li, Abramson, Arlinghaus, et al. Data from qin-breast. *The Cancer Imaging Archive*, 2016.

[27] Du and Vardhanabhuti. 3d-radnet: Extracting labels from dicom metadata for training general medical domain deep 3d convolution neural networks. In *Medical Imaging with Deep Learning*, 2020.

[28] Graham, Perriss, and Scarsbrook. Dicom demystified: a review of digital file formats and their use in radiological practice. *Clinical radiology*, 60(11):1133–1140, 2005.

[29] Heath, Bowyer, Kopans, Kegelmeyer, Moore, Chang, and Munishkumaran. Current status of the digital database for screening mammography. In *Digital mammography*, pages 457–460. Springer, 1998.

[30] Carretero, Isaila, Kermarrec, Taïani, and Tirado. Geology: Modular georecommendation in gossip-based social networks. In *2012 IEEE 32nd International Conference on Distributed Computing Systems*, pages 637–646. IEEE, 2012.

[31] Kundu, Peifer, and Karuppiah. Federated system for centralized management and distribution of content media, August 20 2019. US Patent 10,387,976.

[32] Ciampi, De Pietro, Esposito, Sicuranza, Mori, Gebrehiwot, and Donzelli. On securing communications among federated health information systems. In *International Conference on Computer Safety, Reliability, and Security*, pages 235–246. Springer, 2012.

[33] Sánchez-Gallegos, Carrizales-Espinoza, Reyes-Anastacio, Gonzalez-Compean, Carretero, Morales-Sandoval, and Galaviz-Mosqueda. From the edge to the cloud: A continuous delivery and preparation model for processing big iot data. *Simulation Modelling Practice and Theory*, page 102136, 2020.

[34] Sánchez-Gallegos, Galaviz-Mosqueda, Gonzalez-Compean, Villarreal-Reyes, Perez-Ramos, Carrizales-Espinoza, and Carretero. On the continuous processing of health data in edge-fog-cloud computing by using micro/nanoservice composition. *IEEE Access*, 8:120255–120281, 2020.

[35] Gonzalez, Sosa, Diaz, Carretero, and Yanez. Sacbe: A building block approach for constructing efficient and flexible end-to-end cloud storage. *Journal of Systems and Software*, 135:143–156, 2018.

[36] Xiong, Zhang, Zhu, and Yao. Cloudseal: End-to-end content protection in cloud-based storage and delivery services. In *SecureComm*, pages 491–500. Springer, 2011.

[37] Albrecht, Donnelly, Bui, and Thain. Makeflow: A portable abstraction for data intensive computing on clusters, clouds, and grids. In *2012 SWEET*, pages 1–13, 2012.

[38] Montella, Di Luccio, and Kosta. Dagon*: Executing direct acyclic graphs as parallel jobs on anything. In *2018 WORKS*, pages 64–73. IEEE, 2018.

[39] Deelman, da Silva, Vahi, Rynge, Mayani, Tanaka, Whitcup, and Livny. The pegasus workflow management system: Translational computer science in practice. *Journal of Computational Science*, page 101200, 2020.

[40] Santiago-Duran, Gonzalez-Compean, Brinkmann, Reyes-Anastacio, Carretero, Montella, and Pulido. A gearbox model for processing large volumes of data by using pipeline systems encapsulated into virtual containers. *Future Generation Computer Systems*, 2020.

[41] Wiseman, Schwan, and Widener. Efficient end to end data exchange using configurable compression. *ACM SIGOPS Operating Systems Review*, 39(3):4–23, 2005.

[42] Meister and Brinkmann. Multi-level comparison of data deduplication in a backup scenario. In *Proceedings of SYSTOR 2009.*, page 8. ACM, 2009.

[43] Meister and Brinkmann. dedupv1: Improving deduplication throughput using solid state drives (ssd). In *MSST 2010*, pages 1–6. IEEE, 2010.

[44] K. Miller. Cloud deduplication, on-demand: Storreduce, an apn technology partner: Amazon web services, Mar 2018.

[45] Liu, Yang, Zhang, and Chen. External integrity verification for outsourced big data in cloud and iot: A big picture. *Future generation computer systems*, 49:58–67, 2015.

[46] Chen and Lee. Enabling data integrity protection in regenerating-coding-based cloud storage: Theory and implementation. *IEEE transactions on parallel and distributed systems*, 25(2):407–416, 2013.

[47] Puttaswamy, Kruegel, and Zhao. Silverline: toward data confidentiality in storage-intensive cloud applications. In *Proceedings of the 2nd ACM Symposium on Cloud Computing*, pages 1–13, 2011.

[48] Zhang and Zhang. Secure and efficient data-sharing in clouds. *CCPE*, 27(8):2125–2143, 2015.

[49] Mao, Wu, and Jiang. Improving storage availability in cloud-of-clouds with hybrid redundant data distribution. In *IPDPS 2015m*, pages 633–642. IEEE, 2015.