Carrizales D. et al. (2019) A Data Preparation Approach for Cloud Storage Based on Containerized Parallel Patterns. In: Montella R., Ciaramella A., Fortino G., Guerrieri A., Liotta A. (eds) Internet and Distributed Computing Systems. IDCS 2019. Lecture Notes in Computer Science, vol 11874. Springer, Cham..

DOI: 10.1007/978-3-030-34914-1_45

A Data Preparation Approach for Cloud Storage Based on Containerized Parallel Patterns

Diana Carrizales[1], Dante D. Sánchez-Gallegos[1(✉)], Hugo Reyes[1], J. L. Gonzalez-Compean[1], Miguel Morales-Sandoval[1], Jesus Carretero[2], and Alejandro Galaviz-Mosqueda[3]

[1] Cinvestav Tamaulipas, Ciudad Victoria, Mexico
{dcarrizales,dsanchez,hreyes,jgonzalez,mmorales}@tamps.cinvestav.mx
[2] Universidad Carlos III de Madrid, Madrid, Spain
jesus.carretero@uc3m.es
[3] CICESE, Ensenada, Mexico
agalaviz@cicese.edu.mx

Abstract. In this paper, we present the design, implementation, and evaluation of an efficient data preparation and retrieval approach for cloud storage. The approach includes a deduplication subsystem that indexes the hash of each content to identify duplicated data. As a consequence, avoiding duplicated content reduces reprocessing time during uploads and other costs related to outsource data management tasks. Our proposed data preparation scheme enables organizations to add properties such as security, reliability, and cost-efficiency to their contents before sending them to the cloud. It also creates recovery schemes for organizations to share preprocessed contents with partners and end-users. The approach also includes an engine that encapsulates preprocessing applications into virtual containers (VCs) to create parallel patterns that improve the efficiency of data preparation retrieval process. In a study case, real repositories of satellite images, and organizational files were prepared to be migrated to the cloud by using processes such as compression, encryption, encoding for fault tolerance, and access control. The experimental evaluation revealed the feasibility of using a data preparation approach for organizations to mitigate risks that still could arise in the cloud. It also revealed the efficiency of the deduplication process to reduce data preparation tasks and the efficacy of parallel patterns to improve the end-user service experience.

Keywords: Deduplication systems · Virtual containers · Parallel patterns · Content delivery · Cloud storage

1 Introduction

The volume of data managed by the organizations has been incremented dramatically over the past years [3]. This trend has been mainly produced because the end-users associated with organizations produce and store data in a consistent manner, which produces a data accumulation effect [15].

The outsourcing of data management tasks with cloud providers has become a popular solution for organizations to manage large volumes of data in a cost-efficient manner [13]. Nevertheless, outsourcing data management commonly also means a lack of controls over outsourced data [18]. This could result in critical incidents such as outages, violations of confidentiality, and unauthorized access [1]. Currently, end-to-end and in-house solutions have been proposed for adding properties such as security [11,20], reliability [5,6], and integrity [19] to the contents. This type of solution mitigates and, in some cases, eliminates the effects produced by a given incident that could arise in the cloud.

However, applying these techniques to the data in real scenarios represents a challenge for the organizations as this process is expensive in terms of operations sent to the outsourcing service and, depending on the data volume, it could be a time-consuming task.

The accumulation of data could increase the inefficiency and costs issues associated with preprocessing solutions. For instance, the existence of duplicated data produced by lousy management of documents versions, or accumulated backups performed by the end-users could result in unnecessary processing time associated to cloud operations of such duplicated data.

Data preparation is a technique mainly used in big data scenarios to preprocess and transform data for improving its quality. We propose to apply this technique to the preprocessing tools used by organizations to prevent suffering side-effects from the lack of control of the outsourced data previously described.

In this paper, we present the design, implementation, and evaluation of an efficient data preparation approach. This approach has been designed to achieve two goals: *(i)* to reduce the number of contents or amount of data to be preprocessed, which also reduce the data sent to the cloud and also reduces the costs of the outsource data management tasks; and *(ii)* to execute the preprocessing operations in parallel to improve the efficiency of the data preparation process.

To achieve the first goal, our approach includes a deduplication system that calculates the hash of each file and indexes them to identify data already prepared. It is expected that this system reduces the amount of data to be preprocessed in the data preparation process. Whereas to achieve the second goal, the approach also includes a preprocessing scheme based on parallel patterns encapsulated into VCs [16,17] that improve the efficiency of the application executed in the data reparation process.

A prototype of this approach was developed and included as an intermediary between the users' folders and a cloud storage service. In a study case, real repositories of satellite images and organizational files were prepared to be migrated to the cloud by using data preparation and retrieval schemes built by using our approach.

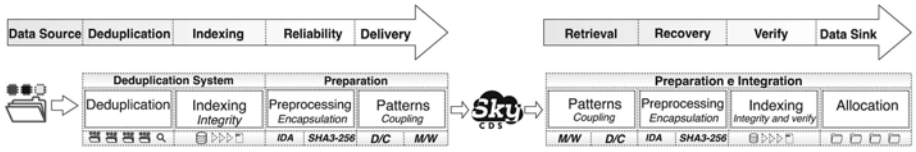


Fig. 1. Conceptual representation of schemes for *data preparation and delivery in cloud* as well as *data decoding and retrieval from cloud*

The data preparation schemes considered in the prototype included applications performing tasks such as compression, encryption, encoding for fault tolerance, and access control. The implementation of these schemes provided to organizations with storage cost-efficiency, security, reliability, and attribute-based access control. The retrieval scheme enables organizations to share prepared data with partners and end-users.

The experimental evaluation revealed the feasibility of using a data preparation approach for organizations for mitigating risks that could arise in the cloud. Retrieval schemes also were evaluated for end-users to share data with other end-users. It also revealed the efficiency of the deduplication process to reduce data preparation and retrieval tasks. It also revealed the efficacy of parallel patterns to improve the end-user service experience.

The paper is organized as follows: Sect. 2 presents the design strategies of a virtual container-based service for cost-efficient data preparation. Section 3 presents the implementation details of a prototype based on the proposed approach. Section 4 describes the experimentation methodology. Section 5 presents the results of the experimentation. Section 6 describes the related work. Finally, Sect. 7 gives conclusion remarks.

2 A Data Preparation Approach Based on Containerized Parallel Patterns

In this section, we describe the design principles of the two components considered in the data preparation approach proposed in this paper. The first one is a deduplication system, and the second one is an engine for the data preparation and retrieval schemes.

Figure 1 shows the conceptual representation of a Client application that *Uploads* contents to the cloud by using data preparation scheme (left). Figure 1 also shows how this client *Downloads* shared contents by using a retrieval scheme (right). As it can be seen, the preparation scheme includes stages such as deduplication and indexing, as well as reliability and delivery. The distribution of the files to the cloud and/or to the end-users is managed by using a content delivery service, called SkyCDS [4], which compresses and encrypts data in-house before transporting them to a multi-cloud storage environment. The retrieval scheme

includes stages such as retrieval of prepared data by using SkyCDS, the recovery from codification performed in the data preparation scheme, the verification of the recovered data integrity, and the allocation of these data in the storage locations of the end-users and partners. Figure 1 will be used to describe each component of the preparation and retrieval schemes.

2.1 A Deduplication System for Cloud Storage

The deduplication service for cloud storage is based on a hash algorithm h (SHA-3-256 [2]), which is used to generate the contents fingerprint $h(C)$ and used to detect duplicates: if $C_1 = C_2$ then $h(C_1) = h(C_2)$. So, h allows identifying replicated contents before sending them to the data preparation scheme. In this system, the hash of each dataset is calculated and compared to the hashes previously discovered by the system (See deduplication system in Fig. 1).

The hashes constitute metadata and the delivery of associated content to the cloud is rejected as these hashes already have been preprocessed in the past. In turn, the hashes that are not found by the deduplication system (*unique data*) are indexed in a hashing metadata database (implemented in MongoDB), and these files are sent to the data preparation system (See indexing in Fig. 1).

In this system, two types of granularity can be used to detect replicated files. The fine-grained control is used by the system to identify, file-by-file, the replicated data. In this type of control, a simple query determines whether a data is replicated or not. In turn, coarse-grained control is used to detect sets of data that are replicated (directories or partitions). In this case, the system implements Merkle trees [12] (also based on hashing) to determine whether a directory already has been preprocessed or not.

In the prototype for preparation data, as showed in Fig. 1 the deduplication system is executed when uploading data in the cloud, whereas the integrity ensuring method is applied to the downloads of contents in information sharing operations for end-users and partners to discover and register alterations in retrieved contents.

2.2 Building Data Preparation and Retrieval Schemes

The second component of our approach is an engine that creates data preparation and retrieval schemes, which are used to preprocessing contents not filtered by the deduplication system. The basic idea of this engine is encapsulating all the components of the schemes created by the approach (including deduplication system, indexing system, and preprocessing applications) into VCs.

The VCs provide the preparation/retrieval schemes with portability property. This enables the schemes to be deployed on different platforms. This means all the software components of each data preparation and retrieval scheme are encapsulated into the VCs with its dependencies such as libraries, packages, OS, and environment variables.

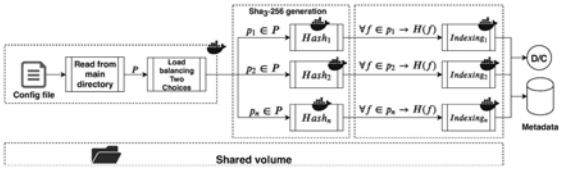


Fig. 2. Deduplication pattern based on VCs.

The VCs are managed in the form of Building Blocks (BBs) that can be coupled by using I/O interfaces such as memory, network, and file system to create a directed graph [16].

In a directed graph, the nodes represent BBs, whereas the edges represent the I/O interfaces used to connect two BBs. A dataflow for either preprocessing or retrieving data is created, when a preparation/retrieval procedure starts by using a directed graph as a guide.

2.3 Parallel Patterns for Data Preparation and Retrieval Schemes

The directed graphs used in this approach enable the approach to build parallel patterns for facing up efficiency issues that arise when preparing and retrieving a large amount of data. In this paper, parallel patterns such as a Manager/worker, Divide&Conquer and fork-join are considered.

Manager/Worker. In this pattern, the *Manager* processes data in phases such as cloning, task definition/distribution, and task execution supervision. In the cloning phase, the Manager creates VC instances of a given stage image included in either preparation or recovery schemes. These clones are used as *worker* instances. In the task definition/distribution phase, the Manager reads, in a recursive fashion, the contents stored in a data source directory DS. In this phase, the manager also creates a list of paths $\mathbf{P} = \{Path_1, Path_2, \cdots, Path_n\}$ and distributes each path to each worker in a load-balanced manner by using the two choices algorithm [10]. In the supervision phase, the *Manager* verifies that all workers are delivering results for the tasks previously assigned to them.

Figure 2 depicts the conceptual representation of the deduplication service as a parallel pattern deployed over VCs. Each worker includes a pipeline that performs the hashing and indexing stages over a sub-list $p_i \in P$ of content paths delivered to the worker by the *Manager*.

The *workers* performs the calculation of the hash $h(\cdot)$ of each content in p_i $(\forall f \in p_i \rightarrow h(f))$ using SHA-3 technique. The hashes calculated are immediately send to a indexing container, called *metadata*. The indexing service finds all the unique hash from the list, saving them in a table $(\mathbf{T}[f_u])$ as $<h(f_u), [f_1, f_2, f_3, \ldots, f_n]>$, where $H(f_u)$ represents the hash of a unique file, and $[f_1, f_2, f_3, \ldots, f_n]$ is a list of paths to the files with the same hash $(h(f_1) == h(f_2) == h(f_3) == h(f_n))$. In order to keep consistency between the

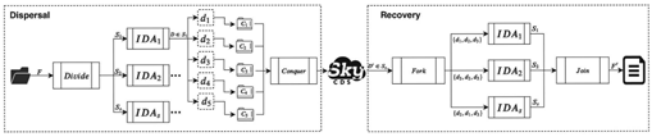


Fig. 3. Reliability and recovery processes by using $D\&C$ pattern.

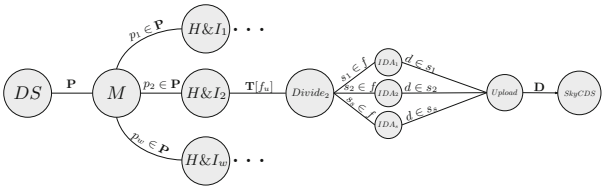
replicated files through the *clients*, the table of hashes is sent to the *metadata* database, where each hash is registered in.

Divide & Conquer. In this pattern ($D\&C$), a software instance called *divide* splits each incoming data into s segments, which are sent to s number of workers previously cloned and launched in configuration time. A software instance called *Conquer* receives the results produced by the workers and consolidates the results to send them to either a next stage in a scheme or to the cloud.

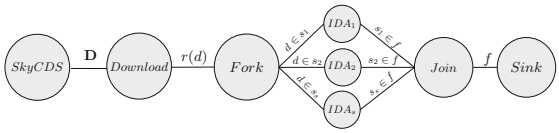
The reliability and recovery stages of the preparation and retrieval schemes are based on the well-known information dispersal algorithm (IDA) [14]. This algorithm splits files into n segments (called dispersal files). In the preparation scheme, the basic idea of using IDA is to add redundancy to each dispersal and to distribute them to n different storage locations. In the retrieval scheme, the idea is recovering the original file by downloading any m segments from n ones created initially and sent them to the cloud (whenever $m < n$). As a result, a file is not sent to the cloud, but its dispersal and the schemes can withstand the failure/unavailability of $n - m$ dispersal files.

In the data preparation scheme, IDA was implemented by using the $D\&C$ pattern. Figure 3 shows an example of this pattern. As it can be seen, *divide* instance splits each data (per processed processed data (d_j)) into c chucks, which are send to c workers. Each worker ($W \in \{IDA_1, \cdots, IDA_n\} \in D\&C$) executes the IDA algorithm to process one segment and produces five dispersal files ($n = 5$); as a result, the pattern produces ($c * n$) redundant portions, which are allocated in shared memory. These portions are retrieved by the *Conquer* from the shared memory and allocates five consolidated dispersal files in each of the five storage locations. In this IDA configuration, the system can withstand the failure/unavailability of two storage locations ($n-m$), which means the retrieval of three dispersal files is sufficient to reconstruct d_j. The redundancy produced by this algorithm is $R = (L/M) * (n - M)$. This pattern produces data parallelism that is quite effective when processing large files (e.g., satellite and health images as well as multimedia and backups).

The recovery stage of the retrieval scheme was implemented as a fork-join pattern. This means a manager retrieves ($m = 3$) three dispersal files from the CDS per each data (d_j) and sends these files to a worker, which reconstructs the original data (d_j). At this point, the patterns deliver d_j to the next stage in the retrieval scheme (e.g., allocation) for end-user can use either store it in a local storage path or send it forward to another CDN or other end-user.



(a) Directed graph data preparation



(b) Directed graph of the data retrieval scheme

Fig. 4. Parallel data preparation and retrieval schemes based on directed graphs.

2.4 Data Preparation and Retrieval Schemes as Directed Graphs

In order to show the interconnection and combination of patterns, we describe the directed graphs of the data preparation and retrieval schemes. Figure 4 shows the data preparation and retrieval scheme represented as a directed graph.

As can be seen, the data are prepared through a set of stages, creating dataflows in pipelines. Bifurcations, producing parallelism, are created when parallel patterns are used in a given stage (see the *Manager/Worker* at deduplication stage and *D&C* at reliability stage in the directed graph of the data preparation scheme).

3 Prototype Implementation Details

The components such as deduplication system, the IDA component, and the hash algorithm h (SHA-3-256) were implemented mainly in C programming. The client implementing the data preparation and retrieval schemes also were developed in C programming.

The integrity verification metadata service has been developed as a web service and a database *Mongo* database deployed in the cloud.

The BBs and the parallel patterns were created by using a framework based on a construction model called Kulla [16]. 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 compression programmed in C and a cryptosystem based on AES and CP-ABE that were programmed in java and incorporated to SkyCDS catalogs.

All the components of the preparation and reliability schemes were encapsulated into VCs created by using the Docker platform.

Table 1. Repositories and IT infrastructure features

	IT features	Set of files			Size (GB)			Duplicated percent	
		Total	Unique	Duplicated	Total	Unique	Duplicated	Files	Capacity
Compute 7	6cores; 24ram	219	130	89	58.29	34.18	24.10	40.63%	41.35%
Compute 8	16cores; 64ram	317465	148147	169318	302.16	146.20	155.96	53.33%	51.61%
Compute 9	12cores; 64ram	126115	43360	82755	300.15	82.54	217.61	65.61%	72.49%
Compute 8	16cores; 64ram	13537	6350	7187	11.63	9.32	2.31	53.09%	11.90%

4 Experimental Evaluation Methodology

The evaluation methodology based on a study case was conducted to evaluate the performance of the prototype of the data preparation and retrieval schemes created by the approach proposed in this paper. The study case was based on real repositories of satellite images and organizational files, which were prepared to be migrated to the cloud by using processes such as compression, encryption, encoding for fault tolerance and access control.

4.1 Metrics

The metrics chosen to evaluate the performance of the prototype were the response time and throughput. The response time represents the time observed by end-users when uploading/downloading contents in the cloud. This time includes the elapsed time spent by a preparation/retrieval scheme to preprocess data (service time) plus the transportation time spent by content delivery network tool. Whereas, the throughput metric represents the number of contents prepared/retrieved per unit time.

4.2 Repositories and Infrastructure

Table 1 shows the features of four servers, which stores the contents of four repositories. The preparation/retrieval schemes were installed in these four servers to preprocess contents of a given directory. SkyCDS was configured to cipher and compress the preprocessed data and then to upload/download them in the cloud.

The first repository (allocated in *Compute 7*) includes 219 files (58.31 GBs). The 95.45% of the files are satellite imagery of *Landsat5*, 277 MB in mean (see data distribution of this repository in Fig. 5a). The second repository (stored in *Compute 8*) includes 317, 465 files (302.16 GB). Figure 5c depicts the distribution of the size of the files of this repository. As it can be observed, the 99.96% of the files of this repository is less than 250 MB. The third repository (stored in *Compute 9*) includes 126, 115 files (300.15 GB). In this repository, the 99.85% of the files have a size of less than 250 MB. This repository was constructed only for experimental purposes, collecting and duplicating files from different data sources.

4.3 Experiments

The data preparation and retrieval schemes were evaluated in two phases. In the first one, was evaluated the costs of the deduplication and integrity verification. The experiments of this phase were performed by varying number of workers in patterns deployed on the stages. In the second phase, reliability and recovery schemes were evaluated, including also deduplication and integrity verification as well as including parallel patterns. The results of these experiments were compared with a system including only the reliability and recovery schemes by using the data form Compute7 repository.

5 Results

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

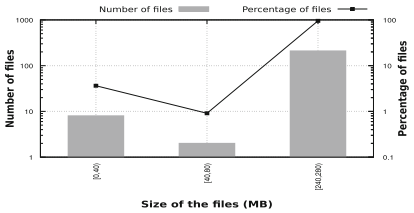
5.1 Analyzing the Costs of the Deduplication and Integrity Verification

Figure 5b and d show, in the left vertical axis, the response time in hours for the stage of deduplication, including analysis and hashing of the files in the data set of *compute 8* and *compute 9*, respectively by using parallelism patterns including a different number of workers (horizontal axis). The right-vertical axis, in both Figures, shows the throughput measured in gigabytes per hour (GB/H).

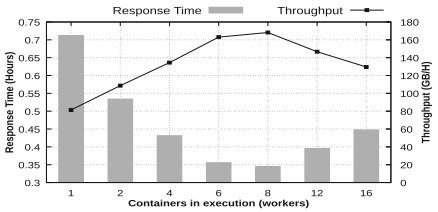
As it can be seen, the task parallelism pattern based on VCs significantly reduces the response time of this stage of the deduplication and integrity indexing, which also reduces the response time of the *Upload/Download operations* performed by the client of cloud storage. This increases the files processed per hour. For instance, the preparation scheme executed in *Compute 7* processed 58.31 GB in 0.71 h (42.72 min) with only one worker. Whereas with six workers completed the very same task in 0.35 h (21.34 min), which means that the parallel pattern with six workers has an improvement of 50.05% with an acceleration of $2x$ in comparison with running only one worker (see Fig. 5b). We also observed that the improvement of the throughput stops when the $NumOfVCs > RealCores$ (number of VCs running in parallel and the number of real processors in a computer) For instance, in Fig. 5b, the response time increase when the pattern was launching 12 workers, improving the performance of the solution by up 44.46% in comparison with running only one worker.

In *Compute 8*, the schemes spent 22.2 h for preprocessing 285 GB by launching only one worker. This produced a throughput of 12.76 GB/H, whereas the M/W with 28 workers completed the same task in 2.2 h. This means an improvement of 89.70% with an acceleration of $9x$ with the 28 workers in contrast with the version of one worker (see Fig. 5d).

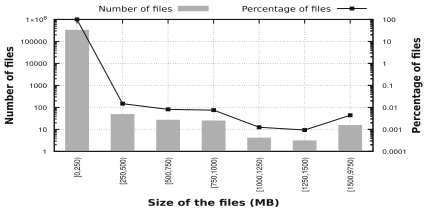
As expected, the task-based parallel patterns based on VCs reduced the response time, increasing the throughput of data preparation and retrieval processes. We observed that this improvement depends on the number of physical cores used by the patterns.



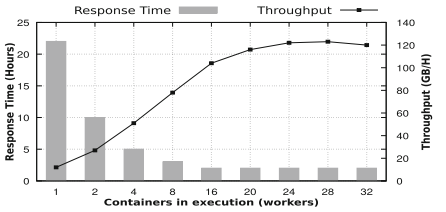
(a) Data distribution of the data set on *Compute 7*.



(b) Response time with a different number of VCs.



(c) Data distribution of the data set on *Compute 8*.



(d) Response time with a different number of VCs.

Fig. 5. Data distribution of the datasets on *Compute 7* and *Compute 8*, and their the response time to process them

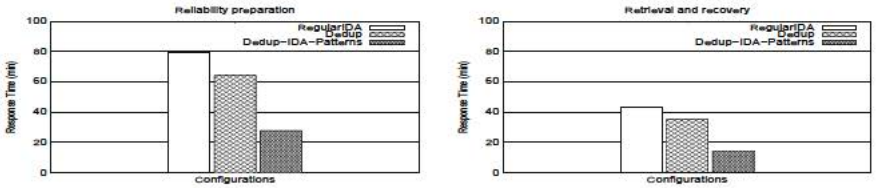
The results of replicated contents discovered and indexed in the integrity verification database are shown in Table 1.

5.2 Analyzing the Impact of Deduplication and Integrity Verification on Reliability and Recovery Stages

The results of the second evaluation phase are described in this section. In this phase we studied the following configurations: *(i) RegularIDA*: This configuration represents the implementation of sequential IDA algorithm; *(ii) Dedup*: This configuration represents the implementation of IDA but including deduplication system; and *(iii) Dedup-IDA-Patterns*: This configuration represents a data preparation scheme (including deduplication and IDA by using D&C pattern).

Figure 6a shows, in the vertical axis, the response time of data reliability preparation and, in the horizontal-axis, the evaluated configurations. The recovery process is shown in Fig. 6b.

As expected, *Dedup-IDA-Patterns* produce the best performance both in preparation and recovery schemes. The response time produced by *RegularIDA* when coding all files (including duplicates) was 79.51 min, the response time produced for *Dedup* configurations was 64.23 min, whereas *Dedup-IDA-Patterns* (by using four workers) was 27.82 min. This means an improvement of 65% for an acceleration of $2.8x$.



(a) Dispersal (b) Recovery

Fig. 6. Response time of the reliability schema for the generation of the dispersals stage (a) and the recovery stage (b).

6 Related Work

In the literature, end-to-end solutions have been proposed for adding properties such as security, reliability, and integrity to the data before transporting them to the cloud.

Morales-Sandoval et al. [11] proposed AES4SEC an end-to-end multi-security service for hybrid clouds, which adds security properties to data by using attribute-based encryption (ABE) and short signatures (SSign). Gonzalez-Compean et al. [5] proposed Sacbe, a BB approach for constructing efficient and flexible end-to-end cloud storage to add reliability property to data.

To provide integrity to data, Xiong et al. [19] proposed CloudSeal, a scheme for securely sharing and distributing data via cloud-based data storage, where the data is encrypted before end-users sharing them with other users. It ensures the confidentiality of contents stored in the cloud. Nevertheless, the major drawbacks of applying these techniques in real scenarios are that organizations have to spend time and resources to manage the operations sent to the outsourcing service, and depending on the volume of the data, this could be cause efficiency problems.

Deduplication techniques to find replicated data have been proposed [7]. StorReduce [9] and Dedupv1 [8] are few examples of this type of solution. There is still an opportunity area to add efficiency property to this type of solutions, which is required for these solutions processing large volumes of data. Moreover, these solutions are commonly developed for a specific platform (OS). This reduces the portability of this type of solution, which is not the case of the approach proposed in this paper.

7 Conclusions and Future Directions

In this paper, we presented the design, implementation and evaluation of an efficient data preparation approach for cloud storage. The approach includes a deduplication system and integration verification method to reduce the number of contents to be preprocessed, which reduce the data sent to the cloud and also reduces the costs of the outsource data management tasks. It also enables end-users to discover alterations of downloaded files. The approach also includes a preprocessing scheme based on parallel patterns encapsulated into

VCs that improve the efficiency of the application executed in the data preparation process. The experimental evaluation revealed the feasibility of using a data preparation approach for organizations to mitigate risks that still could arise in the cloud. It also revealed the efficiency of the deduplication process to reduce data preparation tasks and the efficacy of parallel patterns to improve the end-user service experience. As future work we are working in the evaluation of our approach with other study cases, such as medical imagery and organizational environments.

References

1. Chow, R., et al.: Controlling data in the cloud: outsourcing computation without outsourcing control. In: CCSW 2009, pp. 85–90. ACM (2009)
2. Dworkin, M.J.: SHA-3 standard: permutation-based hash and extendable-output functions (2015)
3. Gantz, J., Reinsel, D.: The digital universe in 2020: big data, bigger digital shadows, and biggest growth in the far east. IDC iView **2007**(2012), 1–16 (2012)
4. Gonzalez, J.L., Perez, J.C., Sosa-Sosa, V.J., Sanchez, L.M., Bergua, B.: SkyCDS: a resilient content delivery service based on diversified cloud storage. Simul. Model. Pract. Theory **54**, 64–85 (2015)
5. Gonzalez, J.L., Sosa, V., Diaz, A., Carretero, J., Yanez, J.: Sacbe: a building block approach for constructing efficient and flexible end-to-end cloud storage. J. Syst. Softw. **135**, 143–156 (2018)
6. Mao, B., Wu, S., Jiang, H.: Improving storage availability in cloud-of-clouds with hybrid redundant data distribution. In: IPDPS 2015m, pp. 633–642. IEEE (2015)
7. Meister, D., Brinkmann, A.: Multi-level comparison of data deduplication in a backup scenario. In: Proceedings of SYSTOR 2009, p. 8. ACM (2009)
8. Meister, D., Brinkmann, A.: dedupv1: improving deduplication throughput using solid state drives (SSD). In: MSST 2010, pp. 1–6. IEEE (2010)
9. Miller, K.: Cloud deduplication, on-demand: storreduce, an apn technology partner: Amazon web services, March 2018.
10. Mitzenmacher, M.: The power of two choices in randomized load balancing. IEEE TPDS **12**(10), 1094–1104 (2001)
11. Morales, M., Gonzalez, J.L., Diaz, A., Sosa, V.J.: A pairing-based cryptographic approach for data security in the cloud. IJISP **17**(4), 441–461 (2018)
12. Ng, W., Wen, Y., Zhu, H.: Private data deduplication protocols in cloud storage. In: Proceedings of the SAC 2012, pp. 441–446. ACM (2012)
13. Plummer, D.C., Bittman, T.J., Austin, T., Cearley, D.W., Smith, D.M.: Cloud computing: defining and describing an emerging phenomenon. Gartner, 17 June 2008
14. Rabin, M.O.: Efficient dispersal of information for security, load balancing, and fault tolerance. JACM **36**(2), 335–348 (1989)
15. Reinsel, D., Gantz, J., Rydning, J.: The digitization of the world: from edge to core. International Data Corporation, Framingham (2018)
16. Reyes, H., Gonzalez, J., Morales, M., Carretero, J.: A data integrity verification service for cloud storage based on building blocks. In: 2018 8th CSIT, pp. 201–206. IEEE (2018)

17. Sánchez, D., Gonzalez, J., Alvarado, S., Sosa, V., Tuxpan, J., Carretero, J.: A containerized service for clustering and categorization of weather records in the cloud. In: CSIT, pp. 26–31. IEEE (2018)
18. Singh, A., Chatterjee, K.: Cloud security issues and challenges: a survey. J. Netw. Comput. Appl. **79**, 88–115 (2017)
19. Xiong, H., Zhang, X., Zhu, W., Yao, D.: CloudSeal: end-to-end content protection in cloud-based storage and delivery services. In: Rajarajan, M., Piper, F., Wang, H., Kesidis, G. (eds.) SecureComm 2011. LNICST, vol. 96, pp. 491–500. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31909-9_30
20. Zhang, J., Zhang, Z.: Secure and efficient data-sharing in clouds. CCPE **27**(8), 2125–2143 (2015)