Abstract—Cloud-based storage services are becoming key for organizations to deliver contents to end-users through the Internet. However, problems such as unavailable service, lost data risks and lack of controls over data management could arise in such a process. This paper presents a framework for building content delivery systems based on continuous workflow (CD-CW). The CD-CW systems split contents into redundant chunks in parallel by taking advantage of multiple cores commonly found in current CPU architectures. The chunks are encoded/decoded, anonymized, and distributed/retrieved to/from different cloud storage locations in a continuous workflow. Our framework allows organizations to automatically build federated cloud storage and publish-subscribe patterns for CD-CW systems. The federated cloud storage allows organizations both to withstand failures scenarios and to preserve autonomy in the data management. The publish-subscribe model establishes controls over the content delivery process and improves the service experience of the end-users. This paper also presents the experimental evaluation of a case study based on satellite images from a space agency. The satellite images are distributed to a federated storage prototype by using a CD-CW agent and a set of organizations, from two countries spanning two continents, get these images by using a CD-CW client.

I. INTRODUCTION

Organizations are struggling to face up the dramatic growth of new data produced, stored, exchanged, and accessed on the Web [11], [16]. In some cases, organizations such as space agencies, hospitals, government instances, news agencies and scientific centers not only are producing huge amount of contents but also they need to distribute them to different communities or population segments through the Internet.

Organizations currently also require to perform backup of these contents for guarantying the continuity of their work flows [11]. Cloud-based approach provides organizations with dynamic, elastic and on-demand storage services [22], which are quite suitable to face up the distribution and backup of large volume of data [1], [5].

However, problems such as unavailable service [3], [4], lost data risks [2] and lack of controls over data management [25] could arise in such a process [13]. Moreover, under the current growing rates, the storing of massive volumes for long periods of time by using public services will have an impact on the long-term cloud storage service costs [28], [30]. Organizations thus must find content delivery and backup strategies that enable them to optimize their data and to solve the problems associated to the cloud approach.

This paper presents a framework for building content delivery systems based on continuous workflow (CD-CW). This framework allows the organizations to automatically build federated cloud storage and publish-subscribe patterns for CD-CW systems.

The federated cloud storage allows organizations to withstand failure scenarios and to preserve autonomy in the data management. The publish-subscribe model establishes controls over the content delivery process as well as reduces the user work associated with the discovery and transfer of new contents.

This paper introduces the following contributions:

- A framework for automatically building CD-CW systems: This framework builds CD-CW systems by using a self-service API. The API builds cloud storage federations and publish-subscribe patterns for CD-CW systems. The federated cells are clusters of cloud-based storage services shared by a set of organizations. The federated cells allows the CD-CW systems to distribute contents on the storage services by using a fault-tolerant scheme. As a result, organizations can solve problems such as unavailable service and lost data risks, which are commonly associated to the vendor lock-in [9], [13].

This framework also allows organizations to build publish-subscribe patterns for CD-CW systems. This enable the CD-CW system to handle operations such as publication and subscription alerts, list of content catalogs as well as push and pull of contents.

- Content delivery optimization: We propose to perform the process of the content delivery as a continuous workflow. The CD-CW systems create this type of workflows by linking the following stages: the publishing and subscription of contents, the acquisition of published/subscribed contents, the transformation of contents into redundant chunks by applying a fault-tolerant scheme, and the transportation of contents from the organization to the end-users through the federated cloud storage. Since CD-CW is based on publish-subscribe patterns, we have designed continuous workflows for push and pull operations. A push workflow starts when an organization publishes a content (publishing-subscription stage). In this case, a CD-CW client reads that content
This paper also presents the experimental evaluation of a case study based on data obtained from the European Space Astronomy Center (ESAC) for the Soil Moisture Ocean Salinity (SMOS mission) (ESA 2008a). We distribute satellite images to a federated storage prototype by using a CD-CW agent and a set of organizations, from two countries spanning two continents, get these images by using a CD-CW client.

The performance of that prototype was compared with a fault-tolerant distributed cloud storage.

II. RELATED WORK

Akamai [14], Coral [15] or Globule [24] are examples of Content Delivery Networks (CDN) [29]. CDN caches small pieces of information and distribute them to locations near who requests content. The final end-user observes a reduction of the latency and overhead in the content delivery process. Our proposal differs from CDN, because it works with relatively small file sizes such as web pages while the CD-CW system has been designed for delivering large files to end-users.

In some solutions oriented to the scientific community, the users need to check for new contents each day by using a web interface. Based on FTP and the bbcp protocols, the contents catalog scheme allows organizations to transfer data by using only 4 simultaneous parallel downloads. Instead, the publish-subscribe model designed for CD-CW allows organizations distribute available contents only to interested users. Moreover, the administrator of a CD-CW system can define the simultaneous number of sockets.

The cloud-based storage services [1], [5] allow organizations to create catalogs of contents based on URLs. The end-users can access the catalog without simultaneous downloads restrictions by using any of the web browsers, synchronizer based on HTTP streams or (S)FTP applications. Nevertheless, studies show that users prefer local storage solutions for managing sensitive data [13].

In addition, Cloud-based storage services are based on a pay-as-you-go pricing model, which apply rates based on the monthly stored contents plus the penalizations stated at the service level agreement (SLAs) contracted [8]. These conditions could lead to long-term costs. For instance, EUMETSAT and EOSDIS transfer around 1 TB of meteorological images per day [10]. This could might press organizations to consider a cost-effective alternative for reducing the service costs at the long-term. Private cloud [12] or a file hosting service (FHS) [18] are alternatives for reducing the long-term costs. In fact, studies show the users favor one-time costs for personal storage rather than monthly charges [20].

FHS has been targeted to those users requiring file storage, who cannot afford neither the expenses of public services, nor the cost of a private cloud. A common FHS is based on URLs, which are constructed on a rather predictable string of characters (explicit user names or sequential identifiers) and therefore, they are exposed to attacks [23].

The CD-CW system can configure either private or public cloud-based federations to distribute available contents to interested users. Moreover, the codification and data dispersion used by CD-CW allows organizations to reduce the required capacity for achieving service continuity. In failure scenarios this method is more cost-effective than the traditional approaches based on replicas.

III. A FRAMEWORK FOR BUILDING CONTENT DELIVERY SYSTEMS BASED ON CONTINUOUS WORKFLOW

In this section we describe how CD-CW systems takes advantage of strategies such as fault-tolerant schemes based on cloud storage federations, continuous workflows based on publish-subscribe patterns and dispersion of contents based on multi-thread codification.

A. Cloud storage federations and publish-subscribe patterns for CD-CW systems

We consider a scenario in which a set of organizations have expressed their willingness to being included in a federated scheme composed of trusted parties, which cooperate to create a private shared storage service. This scenario is quite common when each participant is either a business unit of a larger enterprise, or it is part of a collaborative network [19], [21].

The framework builds a CD-CW system in three phases. In the first phase, the framework builds a cloud storage platform based on storage cell abstractions. In the second, it builds the CD-CW system over the cloud storage cell. In the last phase, the framework builds the workflows and the publish-subscribe patterns for the CD-CW system.

A storage cell abstraction defines a cluster of a set of cloud storage services called workers. The workers are configured as a CD-CW agent and they are in charge of publish-subscribe operations of the CD-CW system (push, pull, catalog list as well as subscriptions and publishing email alerts). The workers are also in charge of handling of a set of cloud storage locations such as containers, virtual lockers o even partitions in which the contents are stored in a redundant manner. The workers are also configured as File Hosting Systems (FHS) to handle contents of the storage locations by using URLs.

The administrator assigns the role of publish-subscribe master to one worker of the storage cell. The master includes a database to synchronize and manage the CD-CW system. This database controls the metadata for federated cloud storage cell, the publish-subscribe registers and the multi-tenancy system for controlling users accounts.

This allows each worker to authenticate and serve user requests by consulting the publish-subscribe master. It also
allows the master to establish controls over the content delivery process.

The framework includes the Only-Global and Local-Global methods to build a federated cloud storage cell. In the only-global method, a storage cell includes either workers from different organizations or storage services from different cloud providers. When either a worker from a given organization or a public provider is unavailable, the master distributes the user requests to the rest of organizations or providers and they reconstruct the contents handled by the worker that is under failure. As a result, the users still can access contents in failure scenarios.

In the Local-Global method, organizations can construct local and global content delivery services. The local service is installed by organizations over a local storage cell constructed by using the only-local method. Once the local storage cell has been built, its master is included as a worker in a federated cell to get access to a global service. As a result, this organization can offer a local content delivery service for their users in an autonomous manner and when this cell is under failure, this organization guarantees the continuity of the content delivery by using the global cell. Obviously, this organization acquires the responsibility to help the federation to serve user requests when other local cell is under failure.

Figure 1 depicts publish-subscribe patterns produced by the the CD-CW system over a federated cloud storage cell built by using the only-global method.

Figure 2 shows how a user constructs a content delivery service based on a federated cloud storage cell by using the self-organizing web application provided by our framework. As we can see, an administrator provides initial information and the API constructs a CD-CW system with this information.

**B. Content delivery optimization by using continuous workflow**

Once the CD-CW system has been created, the organization can start the content delivery service. We have designed this service as a continuous workflow that is applied to push and pull operations.

As we already said, the workflows includes a multi-threading version of the traditional information dispersion algorithm. This algorithm basically splits a given content $|F|$ into $n$ redundant chunks, which must be distributed to $n$ different storage locations. In scenarios in which the user want to retrieve $|F|$ but some of the $n$ original locations are unavailable, the algorithm requires $m$ minimum number of available storage locations to recover it. This means, it is granted that the original content $|F|$ can be reconstructed when $n > m$.

This algorithm can be implemented with different combinations of the $m$ and $n$ parameters. The IDA codification of a given content comes at a price because the size of each resultant chunk is $|F|/m$, which results in a percentage excess of redundancy equal to $(n - m)/m$. Let us consider the traditional IDA implementation with parameters $(n = 5; m = 3)$. In this case, the content $|F|$ is transformed into five chunks and it can be reconstructed by retrieving at least three chunks from three different locations. This means that the combination of parameters $(n, m)$ tolerates up to $(n - m)$ unavailable locations and the system spends an excess of 66.7% on redundancy overhead, which is less than one replica. We thus use this algorithm in the content delivery process for saving storage space in fault-tolerant solutions.

This implementation reduces the size of the content per data transfer as the client is retrieving chunks of size $(|F|/m)$ instead of the whole content.

Since the codification and distribution of chunks produce overhead, we implemented our version of the IDA algorithm to perform the push and pull of chunks in parallel by using TBB library [27].

This multi-threading version improves the performance of these operations by taking advantage of multiple cores commonly found in current CPU architectures. In terms of performance, this version makes feasible for CD-CW to introduce a fault-tolerant scheme in the content delivery process.

In addition, we also implemented HTTP parallel streams by using curl library [7] to deliver contents without restrictions in the amount of simultaneous downloads.

In this implementation the streams are linked to the production and distribution of chunks. This implementation results in a pipeline including four stages: In the acquisition stage, the CD-CW client reads the content $|F|$ from either the local hard disk or a partition. In the transformation stage, the client splits $|F|$ into $k$ anonymized chunks. In the transportation stage,
the client creates $k$ parallel HTTP(s) streams to write/read in/from the cloud storage locations. In the storage stage, CD-CW stores the chunks delivered/retrieved by the streams of the previous stage.

Figure 3 depicts a content delivery process of the content $F$. Figure 3 shows a folder in which the organization stores the contents that they want to deliver to end-users. A multi-thread API allows CD-CW to deliver simultaneous contents by using a dynamic buffer of push and pull processes. Once the multi-thread API starts the content delivery process of $F$ (push in the case shown in Figure 3), this content is sent to the codification and distribution pipeline. As we can see, this process is based on a continuous distribution of chunks $C_1$-$C_5$ with an IDA configuration $(n=5, m=3)$. This means two cloud storage workers can fail and the users still could retrieve the content $F$ by using a CD-CW client.

IV. THE PROTOTYPE AND DESIGN OF EXPERIMENTS

In this section, we describe the experimental evaluation of a case study based on data from the European Space Astronomy Center (ESAC), located at Villafranca del Castillo (Spain), which is in charge of the Soil Moisture Ocean Salinity or SMOS mission [6] from 2009. This mission has generated a vast amount of contents, that will be used to create the first Earth global salinity map.

In this evaluation, our CD-CW agent distributes images of the Sun of SMOS mission in FITS format with a mean size of 44MB to end-users through the Internet. A set of organizations and clients, from two countries spanning two continents, get the contents by using our CD-CW client.

Figure 4 shows a CD-CW prototype in which the images are distributed to a set of organizations from Spain and Mexico. In this scenario, UC3M-Madrid and UC3M-Colme from Spain as well as Cinvestav from Mexico are distributing and retrieving contents by using a federated cloud storage cell.

Figure 4 also shows the phases of the publish-subscribe method. The organization (source) publishes new contents (1), and it can decide either to guarantee a backup of its contents by using the CD-CW push operation (2') or to simply waiting for a download from the CD-CW federation (2). Once the contents are stored in the federation, the master launches a publish operation (3), the users can thus access that content by subscribing it by using a CD-CW client(4). The client retrieves contents and stores them in the user’s device(5). The contents are moved close to the users according to their geographical location.

Table I shows the features of cloud workers and storage services provided by each organization as well as the characteristics of PCs used for handling sources and clients. We used Openstack for building UC3M and Cinvestav clouds.

A. Configurations

The performance of that prototype was compared with fault-tolerant distributed web storage and a distributed cloud storage.
The following configurations were tested to perform the experimental evaluation by using the same publish-subscribe mechanism:

- **DCS**: In this configuration, the end-user can access the contents by using an online distributed cloud storage system or DCS. The organization can withstand the failure of one cloud storage site by using a version of this configuration called **DCS-Mirroring**, which distributes two replicas to the cloud storage. We implemented both DCS versions in the UC3M-Colme, UC3M-Cloud, and Cinvestav organizations.

- **CD-CW**: In this configuration, the organization delivers contents to interested users by using a CD-CW system, which was configured with an IDA combination \((n = 5, m = 3)\). This means the CD-CW client retrieves three chunks each time the end-users subscribe a published content.

### B. Metrics

The response time helps to estimate the user’s and organization’s degree of satisfaction. This time is measured from the publication/subscription moment of a given content until the time point in which the storage service delivers/retrieves that content, meaning that request has been successfully dispatched. This time represents the streaming time, which includes the network round trip latency and the write/read time in the temporal paths. This time also includes the time spent in subscription synchronization. In the case of the CD-CW and Phoenix this time also includes the codification time for redundancy.

### V. Experimental Evaluation

We implemented an API for producing content workloads including both publications of new contents and subscriptions from the clients. The API sends the publications and subscriptions to the CD-CW master, which assumes this artificial load from the clients. We designed a workload scenario to test our proposal in which sun images are published in a sequential manner. In the tests performed in this evaluation, the clients have been configured to randomly subscribe up to 50% of new contents.

We tested **DCS**, **DCS-Mirroring** and **CD-CW** configurations in this scenario.

Figure 5 shows the mean response times observed by organizations when they distribute contents to end-users by using above configurations. Each point in this graph represents the mean response time of ten push operations. This means that 120 contents, with a mean size of 44MB, were sent to the cloud storage by the content delivery configurations.

As can be seen from Figure 5, the response times of **DCS** configuration are better than the **CD-CW** configuration because **DCS** does not produce redundancy during the push operations. Nevertheless, **CD-CW** configuration allows organization to withstand the failure of one cloud site and the source while **DCS** is a backup and it only tolerates the failure of the source.

**DCS-Mirroring** improves the reliability of **DCS** configurations but it introduces a considerable overhead in the response times. Moreover, this configuration sends the whole file and privacy and legal problems could arise. The **CD-CW** system distributes anonymized and encoded chunks. This means only the **CD-CW** agent and client know the locations of the chunks and only these applications can reconstruct the contents. In addition, **CD-CW** only produces up to 66.7% of redundancy overhead while this overhead for **DCS** mirroring this figure is up 100%.

Figure 6 shows the mean response times observed by the end-users from Spain and Mexico when retrieving contents by using **CD-CW**, **DCS** and **DCS-Mirror** configurations.

As can be seen from Figure 6, the response times of pull operations are quite acceptable. The Mexican end-users observe response times similar than end-users from Spain when using **CD-CW** client. This is significant because the source is located in Spain. The **CD-CW** client improves the service experience of the end-users because the **CD-CW** system retains the publication of the contents for the

<table>
<thead>
<tr>
<th>Compute Cores</th>
<th>RAM</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC3M-Cloud</td>
<td>5 Instances</td>
<td>2</td>
</tr>
<tr>
<td>UC3M-Cloud</td>
<td>2 Instances</td>
<td>4</td>
</tr>
<tr>
<td>UC3M-Colme</td>
<td>2 PCs and 4 (17)</td>
<td>4GB</td>
</tr>
<tr>
<td>Workers</td>
<td>8GB</td>
<td>Tamps</td>
</tr>
<tr>
<td>Clients</td>
<td>5 Instances</td>
<td>3(2)</td>
</tr>
</tbody>
</table>

**TABLE I. The features of workers and clients**
Mexican clients while the workers of the Cinvestav cloud retrieves the chunks from the UC3M Cloud. This procedure only takes 4.1 minutes in average and this chunk exchange is performed in automatic and transparent manner for the user. The mirroring inter-cloud procedure performed by DCS-Mirror takes in average 10 minutes. The response time produced by DCS configurations for Mexican clients includes the overhead of retrieving the contents by using the replica allocated in Spain Cloud Storage. The response time overhead observed during the site failure was only 0.7% for CD-CW systems.

The evaluation reveals that DCS-Mirror produces overhead for organizations and improves the response times for end-users. The DCS configuration improves the response time for organizations but it introduces a considerable overhead in failure scenarios and by inter-cloud communications. CD-CW systems offers a trade-off between the benefits observed by organizations and end-users.

VI. CONCLUSION

This paper presented a framework for building content delivery systems based on continuous workflow (CD-CW). This framework allows the organizations to automatically build federated cloud storage and publish-subscribe patterns for CD-CW systems. The federated cloud storage allows organizations to withstand failure scenarios and to preserve autonomy in the data management. The publish-subscribe model establishes controls over the content delivery process as well as reduces the user work associated with the discovery and transfer of new contents.

VII. ONGOING AND FUTURE WORK

We are currently working on an automatic framework for adapting the continuous workflow to the number of cloud storage locations available in an organization by developing different combinations of IDA parameters. We are also defining schemes for load balancing of the storage workers.

ACKNOWLEDGMENT

The authors would like to thank Dr. J. Daniel Garcia for his collaboration in the C++ implementation of the IDA algorithm version for 32 and 64 bit CPU architectures. Grant by CONACYT project 180863

REFERENCES