Contents lists available at ScienceDirect

International Journal of Information Management

journal homepage: www.elsevier.com/locate/ijinfomgt

Research Note

CloudChain: A novel distribution model for digital products based on supply chain principles

Gerardo A. Vazquez-Martinez[a], J.L. Gonzalez-Compean[a,*], Victor J. Sosa-Sosa[a], Miguel Morales-Sandoval[a], Jesus Carretero Perez[b]

[a] *Cinvestav Unidad Tamaulipas Research Center, Cd. Victoria, Mexico*
[b] *Carlos III University, Computer Science Department, Madrid, Spain*

ARTICLE INFO

Keywords:
Information goods
Content delivery
Cloud storage
Security
Supply chain management
Digital product logistics

ABSTRACT

Cloud computing is a popular outsourcing solution for organizations to support the information management during the life cycle of digital information goods. However, outsourcing management with a public provider results in a lack of control over digital products, which could produce incidents such as data unavailability during service outages, violations of confidentiality and/or legal issues. This paper presents a novel distribution model of digital products inspired by lean supply chain principles called CloudChain, which has been designed to support the information management during digital product lifecycle. This model enables connected networks of customers, partners and organizations to conduct the stages of digital product lifecycle as value chains. Virtual distribution channels are created over cloud resources for applications of organizations to deliver digital products to applications of partners through a seamless information flow. A configurable packing and logistic service was developed to ensure confidentiality and privacy in the product delivery by using encrypted packs. A chain management architecture enables organizations to keep tighter control over their value chains, distribution channels and digital products. CloudChain software instances were integrated to an information management system of a space agency. In an experimental evaluation CloudChain prototype was evaluated in a private cloud where the feasibility of applying supply chain principles to the delivery of digital products in terms of efficiency, flexibility and security was revealed.

1. Introduction

Digital products are goods stored, delivered, exchanged and used in digital format. Songs, movies, e-books and magazines are examples of digital products in commercial environments, whereas images, documents and multimedia files are digital products in organizational environments (Berkhout & Hertin, 2004).

The life cycle of digital products (Stark, 2015) includes the transformation from raw to finished/final products through manufacturing/processing stages conducted in collaborative environments by using sharing information tools. The distribution, processing and storage of digital products are keystones in that lifecycle. In commercial environments, expensive and ad hoc solutions are available for companies to build commercial digital products (Nath, Saha, & Salehi-Sangari, 2008; Team, 2014). A solution of this type integrates continuous data delivery, storage tools and content processing applications into a single robust system, usually deployed on dedicated cluster infrastructures where the sharing of information with partners as well as the delivery of finished/final products to consumers are performed by using cloud resources (Gupta, Seetharaman, & Raj, 2013).

In a similar fashion, the distribution, processing and storage of digital information goods in organizational environments is performed through chained procedures where each partner builds either derivative or finished/final digital information products. Spatial and health domains are examples where the digital product lifecycle previously described can be observed.

For instance, in the spatial domain each stage creates derivative products (indexed images) which are acquired and processed by other patterns (distributors) until finished/final products (maps created by using geographical information systems or GIS applications) are produced and available for Consumers (government agencies) for decision-making processes (environment, disaster prevention, climate alerts, etc.). In the health domain, hospitals require diagnostics from a set of medical specialists (e.g., diagnostic about an image taken by a tomograph). In such a situation, the specialists receive images and establish diagnostics from the received images. The images and corresponding

* Corresponding author.
E-mail addresses: jgonzalez@tamps.cinvestav.mx (J.L. Gonzalez-Compean), vjsosa@tamps.cinvestav.mx (V.J. Sosa-Sosa), mmorales@tamps.cinvestav.mx (M. Morales-Sandoval).

https://doi.org/10.1016/j.ijinfomgt.2017.12.006
Received 25 August 2017; Received in revised form 2 December 2017; Accepted 5 December 2017

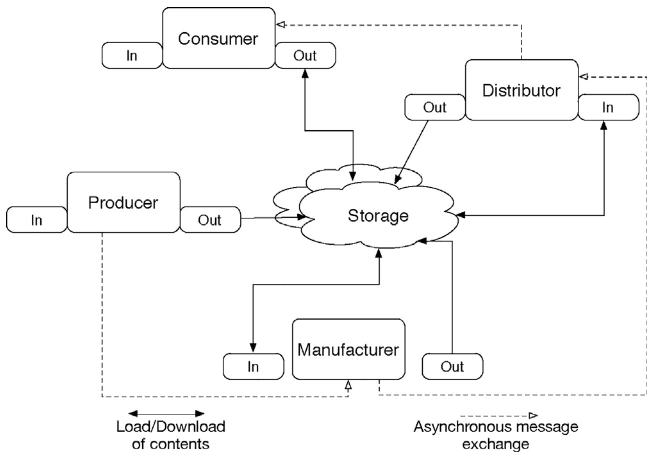

Fig. 1. An example of transformation of digital products life cycle supported by cloud services.

diagnosis are both returned to the hospitals and doctors finally communicate the diagnosis to the patients.

Cloud computing and storage services are cost-effective solutions for organizations to support the life cycle of digital information goods. Synchronization cloud storage applications are useful for producers to share folders with manufactures in a consistent manner (Amazon, 2017a; Bessani, Correia, Quaresma, André, & Sousa, 2011; Gonzalez & Marcelin-Jimenez, 2011; Microsoft, 2017), content delivery tools are useful for the transportation of contents/digital products among transformation Stages (Amazon, 2017b; Gonzalez, Perez, Sosa-Sosa, Sanchez, & Bergua, 2015; OnApp, 2007; Xiong, Zhang, Zhu, & Yao, 2012) (see Load/Download of contents in Fig. 1). The message exchange services (RabbitMQ, 2017) are useful to notify the partners either the arrival or departure of products at each Stage[1] (see asynchronous message exchange in Fig. 1). Cloud services are well suited to enable partners to deliver finished/final digital products to the consumers, which can acquire/download these products in anytime and anywhere manner (Buyya, Ranjan, & Calheiros, 2010; DeCandia et al., 2007; Toosi et al., 2014).

However, the delivery of processed digital products to applications of partners by using cloud services is not trivial. In this type of scheme, coordination and information sharing among a set of partners are both tasks required to support the digital product lifecycle. Moreover, administrators also face up challenges such as removing/adding stages from/to the lifecycle, establishing effective seamless data flow (including message synchronization) and changing any of storage, content delivery or message services by others best suitable for partners or organizations. In this type of scheme, organizations are dependent on the cloud storage and delivery service providers (Opara-Martins, Sahandi, & Tian, 2014). This situation could result in a vendor lock-in, which is critical in service outage scenarios (Brodkin, 2014; McMillan, 2017) where the distribution channels are broken.[2] When an outage is permanent, because of the cloud provider going out of business (Williams, 2013), organizations must perform expensive procedures to migrate data from that provider to a new one (Nasuni, 2015). Moreover, violations of confidentiality (Salveggio, 2004; Sloan & Warner, 2013) and legal issues (Porter, 2012) could also arise when any of delivery, storage or message services are contracted with one single cloud provider (Anitha & Ravikiran, 2014; Chow et al., 2009; Opara-Martins et al., 2014).

In order to reduce dependencies with cloud providers and to minimize the coordination requirements to share digital products, we propose in this paper to manage the distribution, processing and storage required by the life cycle of digital products in organizational scenarios in the form of *value chains*, which connects processing *Stages* deployed by partners (*Entities*) without a centralized control management (Beamon, 1998; Zhou & Benton, 2007).

We present a new distribution model for digital products inspired by lean supply chain principles called CloudChain. In this model, stage software instances (SSI) enable applications of organizations to build distribution channels for delivering digital products to applications over connected networks of end-users and partners. The SSI of CloudChain includes a packing model that creates secured containers for digital products, which are transported through distribution channels by a logistic service by using configurable operations (inbound/outbound logistics).

The administrators can configure CloudChain as a packing and logistic service similar to the service provided by traditional companies that transport packages and/or goods. CloudChain performs associations between folders of local file system with Catalogs (in-box/out-box addresses in this metaphor) and the applications only require to deposit/extract digital products from/to the Catalog folders. CloudChain packs/unpacks products and performs the corresponding delivery to those applications sharing Catalogs through the distribution channels in automatic and transparent manner.

This distribution model produces an alignment of applications of multiple partners that collaborate on upstream processes, downstream processes, or both creating value chains that produce seamless information flows over multiple cloud storage resources. In this model, *supply chain networks* (Beamon, 1998; Zhou & Benton, 2007) are created when organizations establish distribution channels to support alliances with a set of partners. For instance, a supply chain in the aerospace field is built when some companies (manufactures) build derivative products from data collected by several antennas (producers). These derivative products are delivered to different agencies and end-users (consumers). In a similar situation, a value chain is built each time a medical specialist (manufacturer) establish diagnosis for images sent by a hospital (producer) required by several doctors and patients (consumers).

A supply chain management architecture (SCM) was included in the CloudChain for partners and organizations to keep tighter control over the management of distribution channels and digital products through multiple value chains.

The intuitive idea is decoupling applications involved in the life cycle of digital product transformation from the transportation and storage performed by SSI as well as the management of the distribution channels of value chains performed by SCM. This means that the applications have no contact with other applications, but only with CloudChain instances. This feature also allows partners to react to failures and risks in the distribution channels (e.g., changing storage services by other ones). Another interesting feature of value chains built by CloudChain is that the partners only have connection with partners placed in previous and next Stages in a value chain through the CloudChain software instances. This feature allows organizations to preserve sovereignty reducing the side-effects of functional dependencies with Stages and applications deployed in cloud resources by partners.

We developed a prototype based on CloudChain that was deployed on a private cloud. An experimental evaluation revealed the feasibility, in terms of efficiency, flexibility and security, of applying supply chain principles to the distribution of digital products in organizational environments. This prototype is currently being used by a space agency to develop a cooperative delivery platform where value chains are created to transform raster satellite images into derivative products (maps) through cooperative relationships and alliances with a set of partners

[1] commonly are implemented in form of databases or even by using typical asynchronous schemes as email.

[2] Outages produce economic affectations, as shown in a report from Emerson (Anitha & Ravikiran, 2014; Emerson Network Power Independently conducted by Ponemon Institute LLC, 2016; Opara-Martins et al., 2014) showing that organizations suffered, in average, outages for 200 minutes and, depending on the scope of the companies, a minute of outage could cost thousands of dollars.

including government departments and research centers. In this life cycle, both raw digital products (raster images) and derivative products (maps) are delivered to a geo-portal for end-users (consumers) can acquire this type of digital product (Gonzalez, Sosa, Diaz-Perez, Carretero, & Marcelin-Jimenez, 2017).

The outline of this paper is as follows. Section 2 describes existing works related to our solution. Section 3 presents the design principles, architecture, and major components of *CloudChain*. Section 4 describes the prototypes implemented following previous design principles and architecture. Section 5 describes the experiments run to test the system and metrics used to evaluate our proposal. Section 6 presents and discusses the results obtained from the experiments and compares them against other approaches. Finally, Section 7 presents major conclusions of our work and some future lines.

2. Related work

Cloud storage (Amazon, 2017a; Microsoft, 2017) and content delivery services (Amazon, 2017b; OnApp, 2007) based on distributed file systems are solutions available for organizations to centralize the management of digital products, which improves the processes of sharing information. Nevertheless, this type of solution requires the negotiation and coordination of partners to establish a single solution for all participants in the life cycle. Message exchange services (RabbitMQ, 2017) are useful to establish a synchronization among a set of partners, which however must integrate this tool into transportation and storage solutions to create an effective seamless data flow through the cloud. CloudChain reduces the negotiation to a Stage-to-Stage scheme (partner-to-partner), which minimizes the coordination and synchronization requirements to support the exchange of information goods among partners.

In addition, cloud-based solutions produce a functional dependency with the provider (Anitha & Ravikiran, 2014; Chow et al., 2009; Emerson Network Power Independently conducted by Ponemon Institute LLC, 2016; Opara-Martins et al., 2014), which can produce side-effects such as outages, violations of confidentiality, privacy and/or integrity. The transportation of digital products by using multiple cloud storage services (Bessani et al., 2011; Buyya et al., 2010; Gonzalez et al., 2015; Josef Spillner and Schill, 2013; Josef Spillner and Schill, 2014; Xiong et al., 2012) (public, private, heterogeneous or federated) has becoming a popular solution to reduce the side-effects of vendor lock-in as this approach breaks the functional dependency with a single storage/delivery service provider. In this model, the outage/failure of a given service only affects to the Stage that has contracted that service and does not affect the rest of participants in digital product lifecycle, which might contract storage/delivery with another provider. Nevertheless, the synchronization of these services with message service depends on additional implementations for monitoring shared folders (in-box/out-box) of each storage/delivery service created by entities for the exchange of products; as a result, the locations and access processes should be shared among all entities to guarantee a seamless information flow, which reduces the autonomy of partners and could produce security risks in the management of digital products. In turn, in CloudChain each Stage is in charge of processing the messages sent by other Stages, which reduces the overhead by communications. Moreover, the CloudChain software instances take advantage of multi-cloud drivers in the transportation and storage to reduce side-effects from outages of cloud services, which means partners can establish a resilient scheme depending on their cloud resources; as a result, flexibility and efficiency of this solution is achieved as shown in experimental evaluation.

In commercial environments, cluster and cloud-based collaborative applications (Nath et al., 2008; Team, 2014) are available under license schemes and represent a solution for corporations to build digital products through online workflows. The applications are coupled to the delivery procedure through chaining procedures; as a result, the delivery of digital products is contracted with the license of the transformation applications as a whole solution, which requires a dedicated cluster infrastructure. This is affordable and justified for corporations, which require absolute control over the life cycle of the digital products before delivering them to final consumers. In turn, CloudChain is focused on cooperative and collaborative organizational environments where the transformations are distributed among a set of partners. Furthermore, CloudChain enables partners to establish as many value chains as required by business scopes because distribution channels in CloudChain are negotiated partner-to-partner; as a result, the sovereignty of the partners is preserved.

Some companies such as Netflix and iTunes, to our best knowledge, have explored the management of life cycle of digital product as value chains (Nath et al., 2008). However, the design details and implementation are not public.

Solutions that produce continuous content delivery through workflows are available and could be a framework to support the life cycle of digital products. However, this type of solution is focused on software compilation (Jenkins, 2017) and integration of software components to bring the whole software to market quickly (DevOps, 2017). Traditional workflow building tools and web services of specific purpose have been proposed and available to create workflows for the industry of CAD/CAM (Havard Heitlo Holm & Gezer, 2017; Holm, Hjelmervik, & Gezer, 2016). Nevertheless, the generalization of this type solutions is not trivial as licenses and access controls of applications associated to web services must be shared among the partners included into a workflow. In contrast, in CloudChain model each partner is in charge of the licenses of their applications and it does not affect the deployment of value chains as CloudChain decouples the processing performed by the applications from the transportation and storage services of CloudChain.

Cloud computing is a keystone for applications managing real supply chains in companies aiming at improving the exchange of orders (Yun, Cegielski, Hazen, & Hall, 2013), usually focused on the transportation of goods. In turn, CloudChain proposes to take advantage of the supply chains management strategies to applying them to the transportation of digital products through cloud resources in organizational and collaborative environments.

3. The CloudChain digital product distribution model

In this section, we define the concepts considered in CloudChain to establish a model of digital product distribution based on lean supply chain principles.

3.1. Value and supply chains in CloudChain

In CloudChain, an abstraction called *Entity* represents a service for organizations and partners manage data in private manner, whereas the *Stage* abstraction represents a CloudChain service for applications, through the entities, to get access to distribution network build by CloudChain.

Thus, in CloudChain there are two software instances, one for (chained) entities management and another for Stage and distribution channel management. The *distribution channels* are created in CloudChain for the transportation of digital products by chaining Stages, which also creates *value chains*. When CloudChain Stages participate in more than one value chain a supply chain is built.

Fig. 2 shows a conceptual representation of a digital product distribution service based on CloudChain model. In this example three partners have launched CloudChain software instances into their cloud resources. The first instances are called Entity 1, Entity 2 and Entity 3 to administrate value chains, and the second ones are called Stage1, Stage 2 and Stage *n* (depending on if it is required 1 or more Stages by Entity) configured to deliver/retrieve digital products to/from applications (at least one) associated to an Entity, and to perform the transportation of

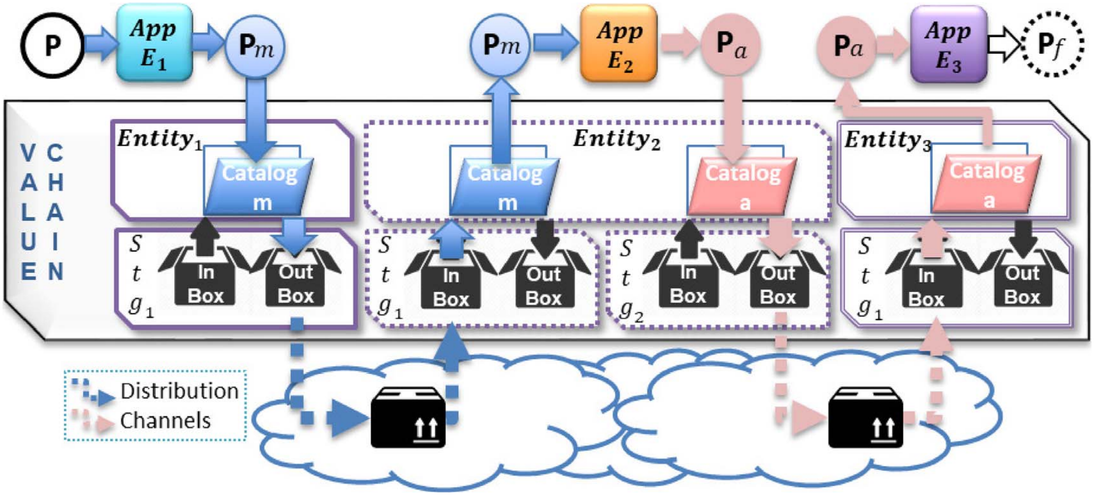

Fig. 2. A value chain in the CloudChain model.

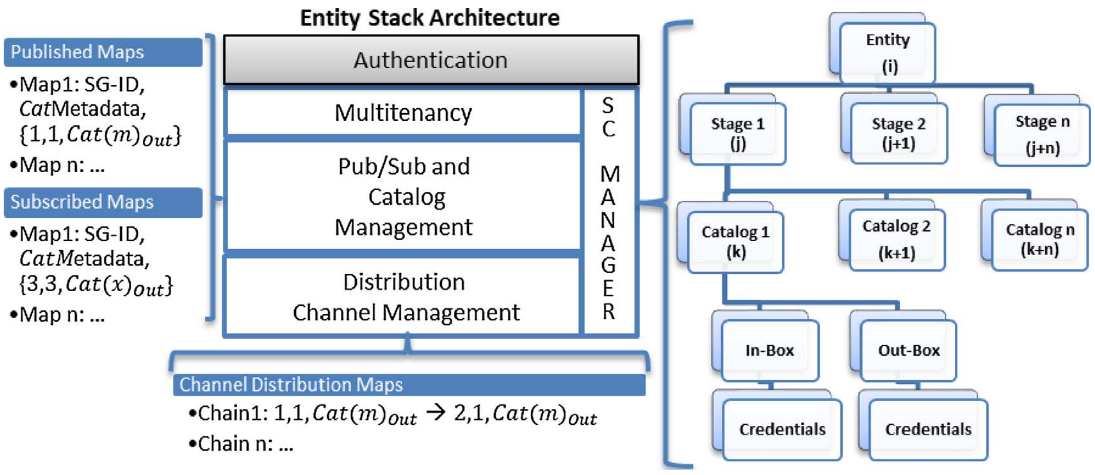
Entity Stack Architecture
Published Maps
•Map1: SG-ID, CatMetadata, {1,1,Cat(m)out}
•Map n: ...
Subscribed Maps
•Map1: SG-ID, CatMetadata, {3,3,Cat(x)out}
•Map n: ...
Authentication
Multitenancy
Pub/Sub and Catalog Management
Distribution Channel Management
SC MANAGER
Channel Distribution Maps
•Chain1: 1,1,Cat(m)out → 2,1,Cat(m)out
•Chain n: ...
Entity (i)
Stage 1 (j)
Stage 2 (j+1)
Stage n (j+n)
Catalog 1 (k)
Catalog 2 (k+1)
Catalog n (k+n)
In-Box
Out-Box
Credentials
Credentials

Fig. 3. Entity stack architecture.

digital products through distribution channels.

A *Catalog* is another abstraction in CloudChain used to manage digital products in the form of groups that are modeled as folders in the local file system. These folders are managed through virtual ports (*in-box* for arrivals and *out-box* for departures of digital products), which are synchronized to cloud storage and/or content delivery services.

To exemplify, consider that Fig. 2 depicts the life cycle of remote sensing data (satellite images). Stage 1 in Entity1 can be associated to an antenna that daily produces satellite images (product *P*). Images are captured from *s* satellites, managed by *v* virtual ports through *c* Catalogs. The application (AppE_1) connected to Stage 1 transforms each *P* by using error correction algorithms into a new version P_m, which is deposited in *Catalog*$_m$, which has been shared with *Entity 2*. The sharing of this Catalog creates a distribution channel between the *out-box* port of Stage 1 in Entity 1 and the *in-box* port of Stage 1 managed by Entity 2, which also results in an automatic transportation of each product P_m produced by AppE$_1$ to Entity 2 through established distribution channel.

When Stage 1 in Entity 2 discovers the arrival of P_m from the distribution channel, moves it to the folder created for *Catalog m* in Entity 2, which means AppE$_2$ gets P_m from local file system to create the derivative digital product P_a (in this example, any of a climate map, image mosaic or overlapped map). This digital product is sent to *Catalog a* managed by Stage 2 in Entity 2 and previously subscribed by Entity 3 (e.g., satellite enterprises, research centers etc). Stage 2 validates P_a and moves it to chain through the *out-box* port. This action creates a distribution channel between *out-box* of Stage 2 in Entity 2 and the *in-box* port of Stage 1 in Entity 3 to transport P_a. When Stage 1 in Entity 3 discovers P_a in the *in-box* port, moves it to *Catalog a* and AppE$_3$ can get it to create finished product P_f, which can be distributed among end-users (e.g., government agencies, universities, news agencies) through either a new distribution channel or other end-point (e.g., Geo-portal, web-pages, cloud storage URL, etc).

3.2. Design principles of CloudChain architecture

The design of CloudChain architecture considers two management blocks based on software instances, one for *entities* and another for *Stages*. The Entity software instance (ESI) enables organizations, partners or individuals to manage Catalogs and value chains, whereas the Stages software instance (SSI) enables applications associated to Stages to get/put digital products from/to distribution channels.

3.2.1. Supply chain architecture: entity stack

The architecture of the Entity software instance includes access and management layers. Fig. 3 shows the components of this two-layer architecture.

The *access layer* includes modules for the management of accounts of administrators and the creation/management of access control lists (ACLs) based on service tokens dynamically assigned to applications and Stages.

The *management layer* enables the authorized users of an Entity

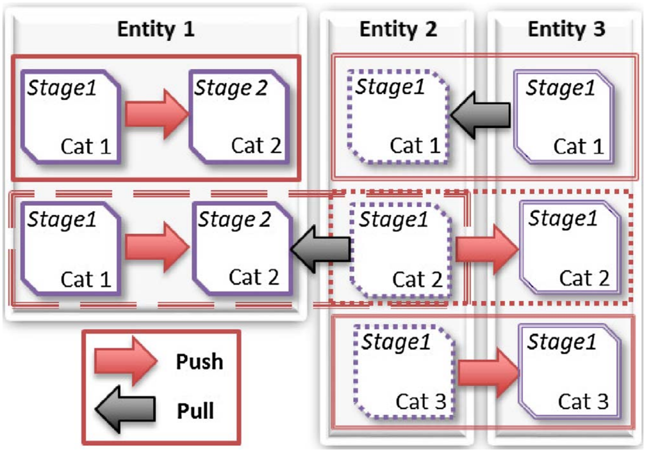

Fig. 4. Planning of value chains.

software to establish controls over Stages and their distribution channels.

A multitenancy module includes functions for entities to create, configure, deploy, remove and list existent Stages. This module enables entities to associate applications to Stages, which isolates the digital products managed by each Stage from other Stages.

The Pub/Sub management module enables Stages to establish controls over the sharing of *Catalogs*. This module includes Publish/Subscribe including *Pub* operations for Stages to publish a Catalog and *Sub* operations for Stages to subscribe a Catalog published by other Stages. The management unit in this module is a map that associates each Entity *i* deploying *j* Stages each managing *k* Catalogs with the corresponding *in-box* and *out-box* ports. Thus, a map is expressed as $\{i, j, k_{in/out}\}$ and represents the ID of each resource managed by a Stage (a Catalog).

Maps are stored in publication/subscription tables. The tuples in a publication table define the group of Stages that can access digital products produced by the Entity, always that the Entity subscribes one of these tuples. The tuples in the subscription table indicates the Stages that deliver digital products to the Entity. In this case, the map includes additional metadata for the Catalog.

Each Map published by an Entity and subscribed by another one represents a chaining of virtual ports between two Stages. This association is used to create a distribution channel. The distribution channel management module registers each valid association for all Stages deployed by an Entity instance (see examples of channel distribution maps in Fig. 3).

The transportation of digital products in the value chain is determined by the maps in the distribution channel table of an Entity. This process can be executed by using either *push* or *pull* operations.

A *push* operation represents an outbound logistic denoted as $M_i \rightarrow M_j$ where M_i is a map in a publication table of Entity *i* and M_j is the map in the subscription table of Entity *j*. The Stage S_j in M_j is in charge of transporting the digital products managed by Stage S_i.

A *pull* operation is denoted as $M_i \longleftarrow M_j$, and indicates that M_i is a map in the publication table of Entity *j*, and M_j is a map in the subscription table of Entity *i*. The Stage *j* is in charge of the transportation of digital products towards Entity *i* at Stage *i*.

The supply chain manager (see SC Manager in Fig. 3) is a transversal component of the Entity stack architecture that integrates the information produced by multitenancy, Pub/Sub and distribution channel modules into a tree that is used to keep control over the Catalogs managed by each Stage. This manager enables modules to share information each other (see the tree of virtual ports shown in Fig. 3). This tree also includes the credentials required by each Stage to transport digital products through distribution channels (e.g., tokes and accounts required by the cloud storage that will be used by a given distribution channel map). In order to clarify the construction of distribution channels, we focus on tables shown in Fig. 3. In the publication table a map M_i $(\{1, 1, Catm_{out}\})$ indicates that Stage 1 of Entity 1 is publishing Catalog *m* through the *out* virtual port, which has been subscribed by Stage 1 of Entity 2 by using a map M_j $(\{1, 1, Catm_{out}\})$ and a push operation. This action produces a distribution channel $M_i \rightarrow M_j$, which means the Entity 1 is in charge of sending data through the distribution channel (see $\{1, 1, Catm_{out}\} \rightarrow \{2, 1, Catm_{in}\}$ in the channel distribution maps in Fig. 3); as a result, all products stored in Catalog *m* by Entity 1 in Stage 1 are transported, in automatic and transparent manner to the In-box port managed by Stage 1 of the Entity 2. Notice that in practice the numbers of Stages, entities and Catalogs are changed by IDs based on large tokens.

3.2.2. Design and deployment of value chains

The administrators of entities can design *Intra-chains* and *Inter-chains*. In an *Intra-chain*, the Stages are deployed by a single Entity, which means the Entity's administrators have the whole control of the life cycle of digital products (over all the Stages in a chain) and can assign logistics for the chaining of all Stages of value chains. In turn, an *Inter-chain* includes Stages managed by different entities. In this type of chains, entities must negotiate the configuration of the distribution channels (e.g., the credentials of Catalogs and the logistic operations).

An *Inter-chain* is deployed when all Stages agree becoming part of a value chain and the configuration files including logistics and maps have been exchanged by the Stages in the chain; as a result, the Stages in the middle of the chain only know the maps of previous and next Stages, whereas the initial and ending Stages only knows information about next and previous Stage respectively.

Fig. 4 shows an example of combinations of Stages, Catalogs and entities created by an Entity's administrator. In this example, Entity 1 created one Intra-chain (between Catalog 1 of Stage 1 and Catalog 2 of Stage 2, which is expressed by the following notation:

$$\mathrm{VC1} = (1,\ 1,\ 1_{\mathrm{out}}) \rightarrow (1,\ 2,\ 2_{\mathrm{in}})$$

In this example, Entity 1 also builds the inter-chain:

$$\mathrm{VC2} = (1,\ 1,\ 1_{\mathrm{out}}) \rightarrow (1,\ 2,\ 2_{\mathrm{in}}) \longleftarrow (2,\ 2,\ 2_{\mathrm{in}})$$

Entity 2 creates three virtual chains:

$$\mathrm{VC1} = (2,\ 1,\ 1_{\mathrm{out}}) \longleftarrow (3,\ 1,\ 1_{\mathrm{in}})$$

$$\mathrm{VC2} = (2,\ 1,\ 2_{\mathrm{out}}) \rightarrow (3,\ 1,\ 2_{\mathrm{in}})$$

$$\mathrm{VC3} = (2,\ 1,\ 3_{\mathrm{out}}) \rightarrow (3,\ 1,\ 3_{\mathrm{in}})$$

Fig. 4 shows that, for instance, Entity 1 shares Catalog 2 with Stage 2 in the value chain (VC2) by using a pull operation. Stage 2 shares this Catalog with Stage 3 through Value Chain (VC2) by push. Please note that Stage 3 ignores that the digital products of Catalog 2 have previously been processed by Stage 1, which ignores the fact of Stage 2 is sharing Catalog 2 with Stage 3.

The notation of value chains is used by SC Manager to accept/reject the processing of digital products in the Stages and enables administrators to design value chains in a straightforward manner.

3.2.3. Orders for the management of digital products in value chains

The CloudChain distribution model, in a similar way to real supply chains, faces up the challenge of conducting an efficient management and exchange of *orders*, which are keystones in the synchronization of the product delivery in a chain. In CloudChain, an *order* is a data structure that keeps meta-information about a given digital product. A Stage only accepts processing a digital product arriving at that Stage with a corresponding and valid order sent by a valid Stage in the value chain (matching with at least one notation controlled by the SC manager). An order includes information about the processing the Stage must perform over a given product, the applications/humans enabled to get access to a digital product to be processed, the transport service token or TST (number tracking), Pub/Sub maps, and the logistics associated to each digital product.

The basic idea is to exchange *orders* between Stages linked by value chain maps to validate the arrival/departure of digital products. In more detail, an order is managed as a Tuple denoted as:

$$O_{\mathrm{Pi}} = \{M_{\mathrm{Pi}}, T_{\mathrm{Pi}}, S_{i,j,k}, L, \mathrm{ST}_{\mathrm{Pi}}\}$$

where M_{Pi} represents metadata about a given product *Pi* such as name and size, T_{Pi} represents the transformation process that is going to be applied to *Pi*, $S_{i,j,k}$ represents the next Stage in the value chain where the processed version of *Pi* must be sent forward in the chain, *L* represents the type of logistic that must be used to transfer orders O_{Pi} and products *Pi*. Finally, ST_{Pi} represents the token that allows Stage to determinate whether both the order (O_{Pi}) and the product (*Pi*) exist or not in the table of value chain maps.

3.3. Stack architecture of Stage management software instance

The Stage software instance is based on a stacked architecture that establishes controls over the arrival and departure of digital products at/from a Stage by using orders as a unit of exchange information. This architecture (depicted in Fig. 5) includes layers such as access, management, order/products processing and data distribution.

3.3.1. Access layer

In this layer (see Fig. 5), an authentication mechanism is in charge of the acceptation/rejection of tokens sent by Stages trying to put/get digital products to/from the virtual ports (in-box or out-box) of a given Stage. The *Front-End* verifies the tokens of external Stages, whereas the *Back-End* is in charge of sending/negotiating tokens to/with another Stages, which listen those requests through the Front-end.

3.3.2. Management layer

This layer includes modules for the management of orders and digital products arriving/departing at/from Stages of a value chain.

An abstraction called *Bucket* is defined to temporarily or permanently store either products or orders and enables Stage instances to manage the flow of orders and digital products through the stack. We defined one type of bucket for the *Stage context* and another one for the *chain context.*

The buckets in the Stage context are deployed on a private space of folders created by Stage software instance in the local file system associated to virtual ports (either in-box or out-box). Thus, only applications associated to an Entity/Stage can get access to the products arriving at this type of bucket (see *In-Box* and *Out-Box* buckets in Fig. 5).

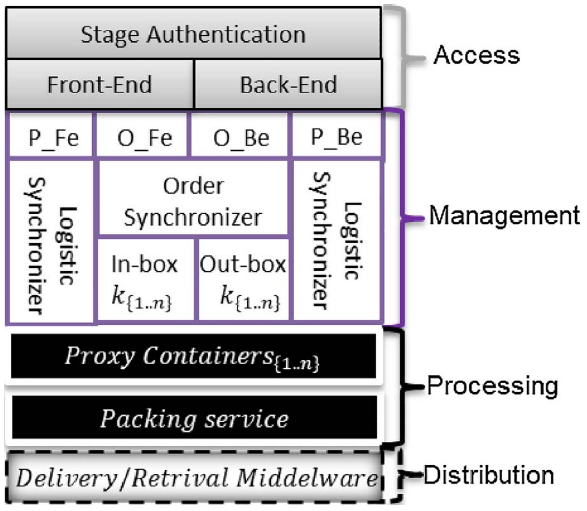

Fig. 5. Stack architecture for the management of orders and products in stage instance.

The buckets in the chain context are temporal folders created for distribution channels (shared with another Stage through virtual port controlled by either Front-end or Back-end). This type of bucket is synchronized with either a cloud storage or content delivery service. The bucket O_{fe} is used to store orders through Front-End, whereas the bucket O_{be} is used to deposit orders leaving a Stage through Back-End. P_{fe} bucket is a similar to O_{fe} but for digital products, whereas P_{be} is similar to O_{be}.

The *order synchronizer* module keeps monitoring the order buckets (O_{fe} and O_{be}) to accept/reject digital products arriving/departing at/from a given Stage. This module extracts orders from corresponding buckets and creates a list of product-order maps, which can be accessed by modules of Stage to verify whether a digital product can be processed or not in that Stage. A Stage only processes products included in that list of maps.

The *logistic synchronizer* module performs three tasks: (i) the monitoring of the product buckets (P_{Fe} and P_{Be}) to discover new digital products to be managed and processed. (ii) sending of queries to the order synchronizer for determining whether a corresponding and valid order has arrived or not for each discovered digital product. In case the order synchronizer finds the corresponding order of a discovered product, it returns the logistic information for the discovered product. Otherwise, a register is added to the log of incidents, which is monitored in the Entity instance to report incidents to administrators as a digital product has arrived at that Stage but there is no a valid order for that product in order buckets. (iii) movement of valid digital products from a context to another depending on the logistic defined in the order. For instance, this synchronizer moves digital products from P_{Fe} (chain context) to In-Box (Stage context) when retrieval operations are performed by Stage, whereas products are moved by the synchronizer from Out-Box (Stage context) to B_{be} (chain) in delivery operations. When the products have been moved to the corresponding context, the Stage can send them to the processing layer.

The administrators can configure the sensing interval for a given workload of digital products. This interval represents the time in which the daemons of the order and logistic synchronizers must wake up to perform corresponding tasks for the management of products and orders.

3.3.3. Processing layer

The main goals of this layer are: (i) the packing of digital products departing from a Stage to other one as well as unpacking products arriving at a Stage. (ii) The reception/delivery of unpacked/packed digital products from/to the Catalogs managed by the Stage. (iii) the reporting to the SC Manager of the Entity software instance the processing of each digital product, so the SC Manager can register each successful or failed operation (either retrieval or delivery).

The *proxy containers*, shown in Fig. 5, were designed for administrators to encapsulate applications in Stages. This non-functional component builds continuous data processing in a value chain. It is useful when the intervention of either external applications or users is not required in a value chain and Stages can automatically perform a processing task (e.g., compression, codification, encryption, water marking etc.). This module is disabled when the external applications (e.g., geographic information system (GIS) and image and multimedia annotation tools) that require the human intervention are using a Stage software instance to retrieve/deliver digital products.

The *packing digital product* module packs digital products and orders going out of a Stage to a value chain and unpacks products coming into the Stage. When the supply chain is used to transport non-sensitive digital products, the packing only adds tracking guides, orders and digital products to the packs. This information is required by Stages to process digital products in a value chain.

In turn, for the management of sensitive digital products, a non-functional component based on the concept of cryptographic digital envelopes (Morales-Sandoval & Diaz-Perez, 2015a; Morales-Sandoval, Gonzalez-Compean, Diaz-Perez, & Sosa-Sosa, 2017; Rosenberg, 2010; Yanez-Sierra, Diaz-Perez, Sosa-Sosa, & Gonzalez, 2015) was included in the packing service for Stages to ensure the integrity and confidentiality of both digital products and orders.

When this security component is enabled in the packing module, the packets are included into an additional pack in the form of a secure digital envelope (SDE). The security component that places packs into SDEs was a realization of DET-ABE (Morales-Sandoval & Diaz-Perez, 2015b).[3]

DET-ABE is realized using the Advanced Encryption Standard (AES) (Miller, Vandome, & McBrewster, 2009) to encrypt content and Ciphertext Policy Attribute Based Encryption (CP-ABE) (Bethencourt, Sahai, & Waters, 2007) to achieve a many-to-many encryption of the content encryption key. The encryption Stage does not need to know the receiver Stages in advance, but encrypts to all those whose attributes satisfy an access policy, thus providing confidentiality and fine-grained access control at the same time. In this way, several different Stage recipients are able to decrypt and then access digital products and the secure sharing of whole Catalogs is feasible as only those authorized entities and Stages with a valid set of attributes could decrypt and recover the AES-key k to launch the AES decryption process over the encrypted packs of products and recover packed products in plain form. The trusted authority required in CP-ABE for the initial setting is the Entity in charge of a Stage. It is responsible of generating a public key PK and master private key MK, both compliant to a given security level. These keys (and also other operations in CP-ABE encryption and decryption) are generated using Pairing Based Encryption (Boneh, 2012), which requires an elliptic curve and pairing setting. The trusted authority is also responsible to generate private decryption keys for Stage software instances, given a set of attributes and MK. DET-ABE supports the three standard security levels of AES, 128 bits (minimum), 192 bits (medium) and 256 bits (high), whereas CP-ABE uses key-length equivalent to AES in order to ensure similar security; as a result, multiple security levels can be applied to the Catalogs managed by Stages depending on requirements of entities (organizations, users and partners).

In summary, the encryption performed in the packing module is carried out as follows:

1. An AES-key (k) of size s is securely generated.
2. Digital product DP is encrypted with AES producing the ciphertext CT_{AES}.
3. The pairing parameters (curve type) associated to s are selected and used to initialize CP-ABE.
4. A policy P is constructed over a set of valid attributes A. The key k is encrypted with CP-ABE, by using A and producing the ciphertext $\text{CT}_{\text{CP}-\text{ABE}}$. The public key PK compliant with the security level s is used during this encryption process.
5. The digital envelope SDE is then constructed by packing CT_{AES}, P, $\text{CT}_{\text{CP}-\text{ABE}}$, and s.

The decryption is performed when the following tasks are executed:

1. The pairing parameters (curve type) associated to s are selected and used to initialize CP-ABE.
2. A private decryption key SK is derived given the list L of Stage decryptor's attributes and MK compliant with s.
3. With SK, CP-ABE decrypts $CP_{\text{CP}-\text{ABE}} \in SDE$, and recovers the session AES-key k.
4. With k, data in plain form is obtained by decrypting $\text{CP}_{\text{AES}} \in SDE$.

[3] Other schemes can be used by just changing the module but keeping the interface, for example the ones reported in Yanez-Sierra et al. (2015) and Morales-Sandoval et al. (2017).

The aim of the digital envelope technique is twofold. It allows using session encryption keys, which make difficult for an adversary to find a key that is used only for a short period of time. It also increases the performance of the entire encryption process because symmetric ciphers are faster than the public key ones, specifically when large and variably sized amount of digital products are encrypted.

3.3.4. Distribution layer

This layer includes a delivery/retrieval middelware based on tools for the connection and synchronization of virtual ports in a Stage to/with multiple cloud storage services. The basic idea is to synchronize buckets of Front-End and Back-End (B_{fe} and B_{be}) to a cloud storage or content delivery service defined by administrators by using either application programming interfaces (API) or web services. This feature enables Stages to add/remove storage services to/from in-box/out-box interfaces in a transparent manner. The services associated to the buckets can be changed by others on the fly whenever this middleware has the API of the new external cloud services and credentials are registered in the Entity software instance.

3.3.5. Digital product tracking

SC tracker is a third party non-functional component for administrators or authorized users to invoke tracking and tracing processes to follow up digital products in supply chains.

This component produces a guide token (G) for all products of a Catalog monitored by a tracking and tracing process. The Stages must request G tokens to the SC Tacker, add them to digital product packs of the Catalogs being monitored by a tracking and tracing process. The Stages report the status of each event in the digital product lifecycle managed in that Stage to the SC tracker by using G token.

The users that invoked a tracking and tracing process can get a report created by SC Tracker from the registers performed by Stages participating in such an process. This interactive report describes the transformations of digital products through a value chain as well as statistics about service times for each operation performed with a product in each Stage.

The report includes tuples containing a generic token to identify and to anonymize Stages, the status reported by each Stage, the type of operation performed by Stages (anonymized by another token) and arriving/leaving time for each event reported by Stages. The agreement achieved by the entities participating in a tracking and tracing process includes a list of Stages that can get service tokens and users accounts that can get access to reports through the SC tracker service. When users are correctly authenticated and the tracking is non-sensitive, the registers are de-anonymized, a report is built and delivered to authorized users in the form of object that can be observed/downloaded in/from the web page of SC tracker. In case of tracking sensitive digital products, the report is de-anonymized and added to a SDE pack with the policy defining the users having the rights to decrypt it. When the SDE is unpacked by a Stage instance, the users can see the tracking report (this feature requires Stages to enable the security functionality in the pack/unpack service). The statistics of reports built by SC tracker allows the Entity's administrators to detect bottlenecks in a chain.

The tracker service can have the following status: (i) *Init*: produced when a Stage requests for a guide to follow up a given product through a chain (see procedure 1 in Fig. 6). (ii) *Product arrival*: registered by the Stage that receives a digital product (see procedure 2 in Fig. 6). (iii) *Product departure*: registered when a Stage sends a processed product to another Stage (see procedure 3 in Fig. 6). (iv) *Delivery/retrieval*: registered when the Stage invokes the SC tracker in a chain to put/get a digital product to/from a chain (see procedure 4 in Fig. 6).

4. CloudChain prototype: implementation details

In this section, we present implementation details of a prototype for the distribution of digital products based on the supply chain

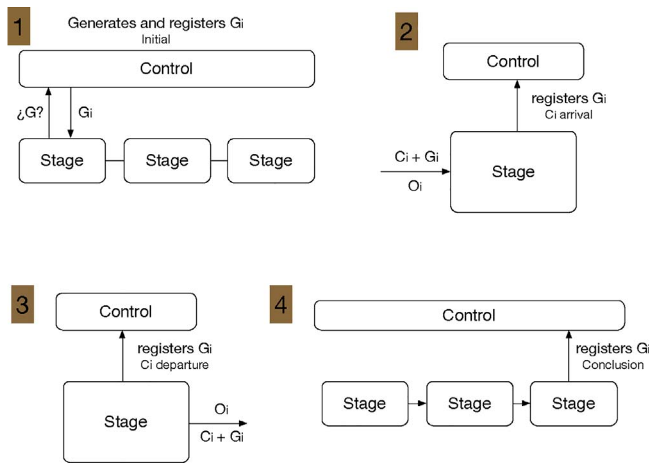

Fig. 6. States of a product tracking guide G_i.

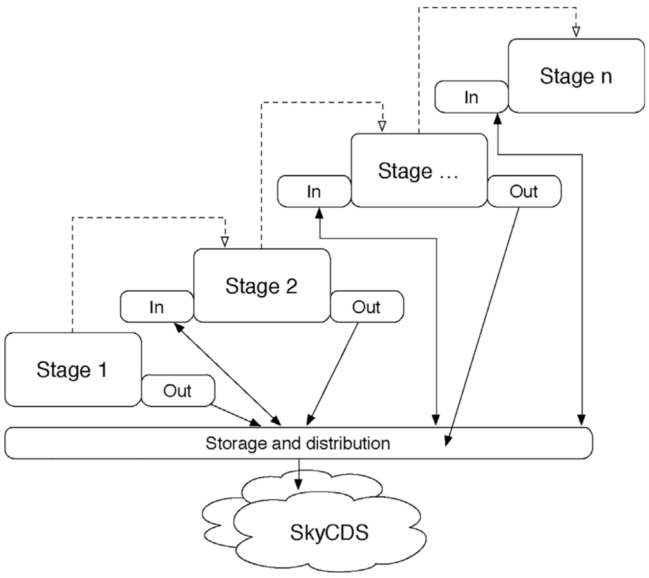

Fig. 7. Conceptual delivery service based on CloudChain.

architecture of CloudChain. This prototype is being deployed on the cloud and cluster infrastructure of a space agency and partners to manage the life cycle of satellite images.

The distribution service of digital products included in the CloudChain prototype was implemented by using cloud images deployed on a private cloud built with Openstack Mitaka (OpenStack, 2015). By image we refer to a virtual machine including a virtual disc with an operative system ready to boot with CentOS 7 and software dependencies. An image was prepared with all the components of a Stage and Entity software instances, an image of a bot that launches workload to the value chains (provider) as well as an image of a multi-cloud storage service for the storage and distribution of digital products. The images were cloned to launch as many cloud instances from a same image as required to conduct the experimental evaluation.

Table 1 shows the image characteristics (Seeds), the number of instances launched in the cloud and the features of each instance (ports, services, and hardware characteristics).

The storage and distribution instances were used for deploying a fault-tolerant, muti-cloud storage and content delivery service called SkyCDS (Gonzalez et al., 2015), which tolerates two cloud site/resources outages/failures by using dispersal information. The virtual ports of Stage software instances in CloudChain were synchronized with storage locations managed by SkyCDS. It is important to note that this cloud storage solution (SkyCDS) can be changed in CloudChain by any solutions based on synchronization (dropbox, azure, s3), Content delivery networks (cloudfront, etc) or a grid platform (Foster, 2005) available in the market. In order to change SkyCDS by a most suitable transportation and storage service meeting partners requirements, an administrator only requires adding the I/O and authentication functions of the API for these services to the SC manager of the Entity software instance.

A conceptual representation of CloudChain service prototype is shown in Fig. 7.

5. Evaluation methodology

In this section, we describe the methodology to evaluate the performance of the CloudChain prototype in a real scenario. The evaluation is performed in two phases. The first one, called *stand-alone scenario*, evaluates the performance of Stages whereas the second one, called *value chain scenario*, evaluates the performance of the digital product life cycle as value chains.

In the second evaluation phase, the performance of CloudChain prototype was also compared to a distribution service that we called $MS_{Centralized}$, which was built with web services (message exchange and processing applications). The SkyCDS was re-used by $MS_{Centralized}$ service to produce a fair comparison. The variations in these experiments include the way in which orders are exchanged among partners and the costs of enabling the security functionality in the Packing/Unpacking service of CloudChain prototype. Table 1 describes the images of this service, which includes an image in charge of the message exchange web service.

A conceptual representation of $MS_{Centralized}$ service is shown in Fig. 8. As it can be seen, the web services send orders to the message service, whereas SkyCDS (Gonzalez et al., 2015) is used for the transportation of digital products and to synchronize folders used by applications of partners. A client application keeps monitoring the shared folders and sending orders to the message service each time a new product arrives at the synchronized folders.

It is important to note that, having as use case the satellite data management, we performed a comparison of CloudChain against an own solution, which was implemented by using traditional web service tools for file transfer, message exchange and cloud storage management. A solution as that is common, in spatial data management scenarios, to support the information distribution, processing and management of digital products lifecycle. We are unable to compare CloudChain with an external existing solution because, to the best of our knowledge, the only two possible and similar solutions, one built by Netflix and another by Itunes, are closed systems, unfeasible to be adapted and evaluated under the same application domains.

Table 1
Features of images and instances used in the digital product distribution service based on CloudChain model.

Seed name	Instances	Services	Ports	Instance configuration (flavor)
Stage	6	Java 8, PostgreSQL, Apache 2, Php5	–,5432,80,–	1 VCPU, 2 GB RAM, 20 GB
Provider	1	Java 8	80	1 VCPU, 2 GB RAM, 20 GB
Storage and distribution (SkyCDS)	5	Java 8, PostgreSQL, Apache 2, Php5	–,5432,80,–	2 VCPU, 2 GB RAM, 40 GB
Messaging service	1	Java 8, PostgreSQL, Apache2, Php5	–,5432,80,–	2 VCPU, 2 GB RAM, 40 GB

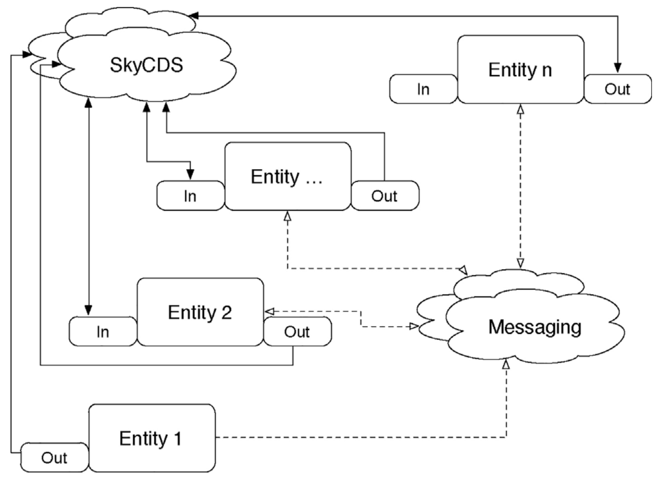

Fig. 8. Conceptual representation of digital product distribution service based on centralized messaging service ($MS_{Centralized}$).

5.1. Metrics

The performance metrics in a real supply chain are key for decision-making processes, so they are for CloudChain. In this section, we describe the metrics considered in CloudChain to measure the performance of Stages, value chains and supply chains.

5.2. Stage metrics

The service time of a Stage (ST_{st}) is calculated by the following formula:

$$\mathrm{ST_{st}} = \sum_{i=1}^{n\mathrm{VC}} \mathrm{ST_{pm}}$$

where ST_{pm} represents the time required by a Stage to manage digital products through the stack architecture (both order and product) for n_VC value chains. ST_{pm} can be calculated by the following expression:

$$\mathrm{ST_{pm}} = \mathrm{ST_{os}} + \mathrm{ST_{ls}} + T_m + \mathrm{TC}$$

where ST_{os} represents the time required by the order synchronizer to recover and store the order (*os*) from/to the order buckets. ST_{ls} represents the service time of the logistic synchronizer to move digital products among the buckets. T_m represents the manufacturing time required in the processing layer to acquire and process a product. T_m depends on the configuration used in a Stage to process the digital product (either an external application or an application encapsulated into containers of the processing layer). *TC* represents the communication time between the daemons of the synchronizers and it is calculated by:

$$\mathrm{TC} = \mathrm{TC_{os}} + \mathrm{TC_{ls}}$$

where TC_{os} represents the elapsed time in which an order is deposited in the order bucket (O_{fe} or O_{be}), and TC_{ls} is similar to TC_{os} but for digital products. $ST_{st} = ST_{pm}$ when a Stage is only associated to one single value chain.

The Stage response time (RT_{pm}) for the management of orders (products included) considers the service time ST_{os} and the time spent of propagation of orders and digital products through virtual ports (*In-box/Out-box*):

$$\mathrm{RT_{pm}} = \mathrm{ST_{pm}} + L_{\mathrm{ini}} + L_{\mathrm{out}}$$

where L_{ini} is the time spent by inbound logistics operations (packing/unpacking of orders/products coming into a Stage), whereas L_{out} is the time spent by outbound logistic operations (orders/products going out from a Stage). $L_{out} = 0$ in the last Stage.

The Stage response time RT_{st} includes the sum of the RT_{pm} for all orders served by a Stage:

$$\mathrm{RT_{st}} = \sum_{i=1}^{\mathrm{VC}} \mathrm{RT_{pm}}$$

$RT_{st} = RT_{pm}$ when a Stage is only associated to one single *VC*.

5.3. Metrics for value and supply chains

The service time of a value chain ST_{VC} is calculated by the sum of service time of all Stages (ST_{st}) by using the following formula:

$$\mathrm{ST_{VC}} = \sum_{i=1}^{\mathrm{st}} \mathrm{ST_{pm}} \quad \text{with st} \geq 2$$

ST_{VC} does not include the time for transportation and storage of digital products, which is calculated by the response time of a value chain:

$$\mathrm{RT_{VC}} = \mathrm{ST_{VC}} + \sum_{j=1}^{\mathrm{st}-1} \mathrm{Pr}_{o/p}, \quad \text{with st} \geq 2$$

where $Pr_{o/p}$ represent the propagation time of orders/products of $st - 1$ Stages in a chain, which can be calculated by the following equation:

$$\mathrm{Pr}_o = T_{\mathrm{td}} + T_{\mathrm{to}}$$

where T_{td} represents the transportation time from the Stage to the Data distribution layer, whereas T_{to} represents the time spent in the transportation of data between two Stages in a chain by the cloud solution chosen for the transportation of products.

The service time of a supply chain (ST_{SC}) is calculated by the sum of the service time of all Stages (ST_{st}) for all value chains (*VC*) deployed by organizations and partners. This metric is calculated by the following formula:

$$\mathrm{ST_{SC}} = \sum_{i=1}^{n} \mathrm{ST_{VC}}$$

The supply chain response time is calculated by the following formula:

$$\mathrm{RT_{SC}} = \sum_{i=1}^{n} \mathrm{RT_{VC}}$$

where $RT_{SC} = RT_{VC}$ and $ST_{SC} = ST_{VC}$ when a supply chain (*SC*) only includes one single value chain (*VC*).

6. Performance analysis and results

In this section, we describe the experiments performed for the evaluation scenarios previously defined and discuss the results achieved in each scenario.

6.1. Stand-alone scenario: a performance evaluation at stage level

This scenario is focused on the performance characterization of the Stage software instance. A bot software called *order generator* created workloads of orders and digital products that were sent to a Stage software instance of the CloudChain prototype, which served each workload as a real load sent by authorized users. Two groups of experiments were performed in this scenario.

For the first group of experiments, the order generator launched workloads to the Stage instance varying the amount of value chains associated to that Stage from 1 to 100 value chains (see workflows in Fig. 9). The demons for the synchronizers of the Stage were configured with two sensing interval policy; the first one wakes up daemons in a time interval of one second, whereas the second considers intervals of ten seconds. In all these experiments, the size of digital products was 10 MB.

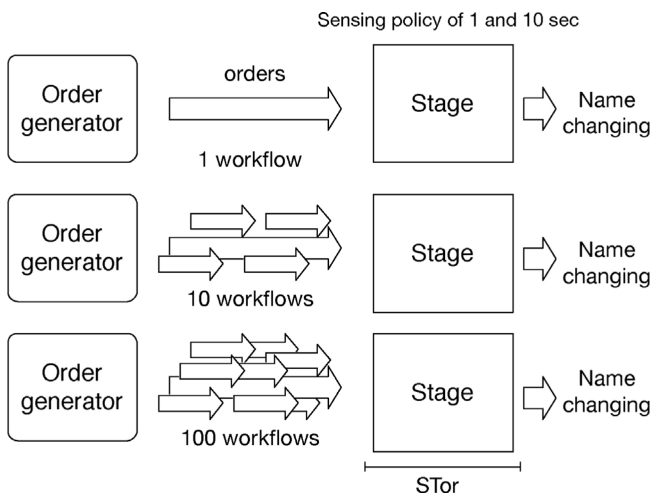

Fig. 9. Stand-Alone evaluation scenario.

For the second group of experiments, the order generator created workloads varying the size of the products from 1MB to 1GB to observe the impact of the size of digital products on the Stage performance.

We executed each variation of the experiments 31 times and captured each component of ST_{pm}. Then, the median of ST_{pm} and each of its components was normalized.

6.1.1. Analysis of Stage performance for the stand-alone scenario

In this section, we present the results of the experiments for the stand-alone scenario.

Fig. 10 shows, in vertical axis, the service time ST_{pm} of two configurations of the Stage software instance evaluated in this scenario each using a different sensing interval (*1 Sec Sensing Policy* and *10 Sec Sensing Policy*). The two Stage configurations were evaluated attending different number of value chains (horizontal axis). As expected, the more the number of value chains managed by a Stage, the more the increment of ST_{pm} for both configurations. We also observed that this behavior is from linear to exponential, which can be observed in Fig. 10 for a sensing interval of 10 seconds when a Stage serves more than 10 value chains in concurrent manner.

In these experiments, we also observed that the less sensing interval, the more the time spent by synchronizations in communications, whereas the more digital products arriving at a Stage, the more the time spent by logistic synchronizer for managing the buckets. The combination of reduced sensing interval with concurrency in the number Value Chains produce an effect of accumulation of digital products in Stage queues, which increments the service time of the Stage.

In order to analyze this accumulation effect of digital products in a Stage, we focused on the components of the ST_{pm} metric, which includes the service times of each demon (ST_{os} and ST_{ls}), the communication time between each demon (TC) and the processing service time by the transformation of content manufacturing (T_m).

Fig. 11 shows, in vertical axis, the service time of each component considered by ST_{pm}, whereas different configurations of the Stage attending different value chains with different sensing policy ({1,10,100} VC{1,10}Sec) are shown in horizontal axis. As expected, the communications (TC) established between order and logistic synchronizers as well as the management of digital products among buckets performed by logistic synchronizer (ST_{ls}) represent both the major portion of ST_{pm}.

Fig. 11 shows that, for Stage participating in few value chains (See *1VC1Sec, 10VC1Sec* and *1VC10Sec, 10VC10Sec*), the communications time between daemons (TC) is higher than that the time produced by the logistic synchronizer (ST_{ls}). The communication time (TC) of 1VC1Sec and 1VC1Sec is higher than 1VC10Sec and 10VC10Sec because these configurations send more requests to the order synchronize in less time. The key metric for a Stage participating in large number of value chains is ST_{ls} because the logistic synchronizer moves more digital products among the buckets of the stage (See ST_{ls} for *100VC1Sec and 100VC10Sec* in Fig. 11). When the logistic synchronizer daemon of the Stage 100VC10Sec wakes up in a concurrency scenario, this configuration discovers more digital products in the bucket system than the Stage 100VC1Sec. This means that 100VC10Sec reduces the communication costs (See ST_{ls} 100VC10Sec) but accumulates products to be processed in concurrent manner in the bucket system (See ST_{ls} 100VC10Sec), which delaying the processing of digital products and affecting the ST of the stage (ST_{pm}). The logistic synchronizer daemon of 100VC1Sec manage less digital products per unit of time than 100VC10Sec avoiding the accumulation of products in the bucket system, which not only improves ST_{ls} but also the service time of the stage ST_{pm} in comparison with 100VC10Sec.

We can suggest that, based on these results, a sensing interval of one second is suitable for scenarios with high concurrency degrees, whereas a sensing interval of ten second is suitable for scenarios where concurrency of value chains or arrival of digital products is low.

It is important noting that the processing layer in the experiments of this scenario only performs the task of anonymizing the name of digital

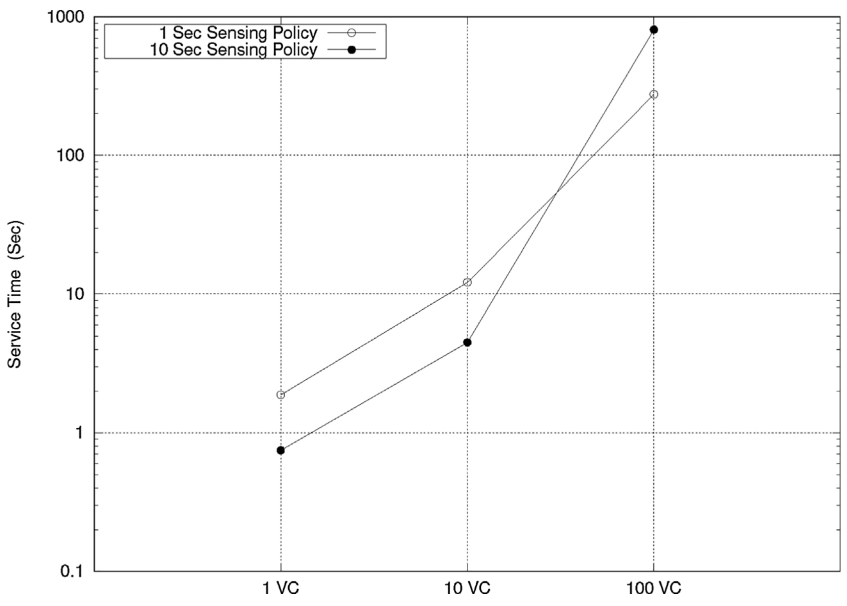

Fig. 10. Service time produced by different sensing intervals varying the number of value chains.

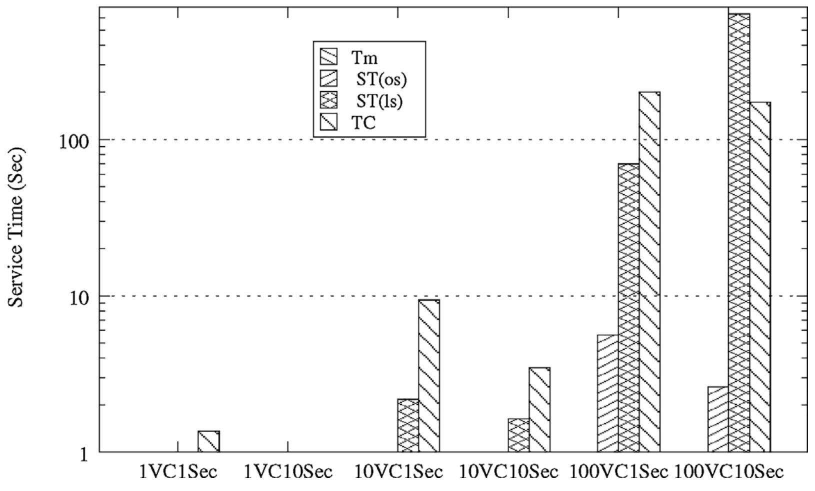

Fig. 11. Service time broken down into components for stage configurations attending different value chains with different sensing policy ({1,10,100}VC{1,10}Sec).

products (which represents P_m); as a result, T_m is not significant. We made this decision to reduce the time of experimentation, to simulate concurrency scenarios (the number of value chains automatically represents the number of concurrent orders/products served by a Stage) and because T_m depends on a component not included in the CloudChain architecture (external applications or applications encapsulated in the Stages by Entity's administrators).

Fig. 12 shows the response time (vertical axis) of last group of experiments in which the size of the digital products is incremented from 1 MB to 1 GB (horizontal axis in ten log basis). In these experiments, the secure version of packing service was enabled/disabled (*ST _ Sec/ ST _ NoSec*) to observe the impact of the size of digital products on the Stage performance.

As it can be seen, the performance of CloudChain Stages does not depend on the digital product size when the security component is disabled in the packing service. But as expected, the performance of a Stage is affected in proportion to the digital product size when the security component is enabled.

6.2. Value chain scenario: a performance evaluation at chain level

In this evaluation scenario, we compared the performance of value chains built by the CloudChain prototype with a content delivery technique based on a centralized messaging service (called $MS_{Centralized}$; see the prototype description of this solution in Section 5).

For the CloudChain prototype, the workload generator bot created one single value chain and introduced orders to the initial Stage to automatically create a seamless flow of products among the Stages. In the case of the $MS_{Centralized}$ prototype, the bot created one single workflow and messages for a set of web services included in that workflow.

The Stages in the case of CloudChain and web services in $MS_{Centralized}$ prototypes were added one at a time (from one to four). The time for the communication between the message service and the web services was one second, which also was used to configure the sensing policy of CloudChain synchronizers. Fig. 13 shows a conceptual representation of the experiments performed in this scenario.

Tasks of compression and decompression were performed in automatic manner by applications encapsulated in the processing layer of CloudChain and web services of the $MS_{Centralized}$ prototype. For both prototypes, the size of digital products was 1MB and the metrics RT_{VC} and ST_{pm} were captured in these experiments.

It is important noting that CloudChain prototype considers five Stages for the management of orders and digital products, whereas $MS_{Centralized}$ prototype considers five nodes of web services and another node for the exchanging of orders among web service nodes (one more cloud instance than *CloudChain* prototype). The transportation of digital products in both prototypes was performed by the same content cloud

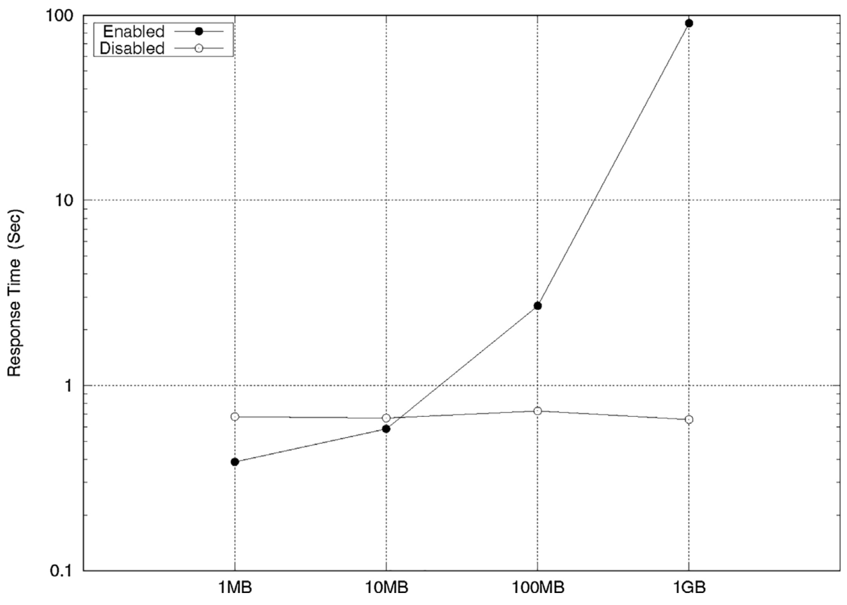

Fig. 12. RT_{st} using different size of digital products with the security function enabled and not enabled.

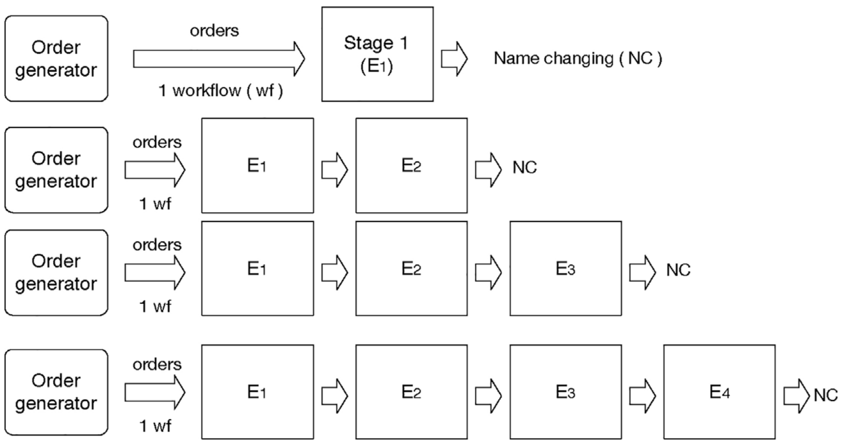

Fig. 13. Conceptual representation for the evaluation of value chain scenario.

delivery and storage service (SkyCDS). This allowed evaluating the performance of the method of exchange of orders and the method for securing digital products, which was activated for these experiments. Each experiment was performed 31 times to normalize the median of RT_{VC} and ST_{VC} captured during these experiments.

6.2.1. *Analysis of the impact of the number of stages on value chain performance*

This section shows the results of the experiments for the value chain scenario.

Fig. 14 shows the RT_{VC} (vertical axis) produced by CloudChain and $MS_{Centralized}$ prototypes depending on the number of the Stages/web services in a value chain/workflow (horizontal axis).

As it can be seen, the RT_{VC} produced by $MS_{Centralized}$ prototype grows with the number of Stages more than the RT_{VC} of CloudChain that follows a linear growth. The performance of $MS_{Centralized}$ is affected by the messages and content delivery synchronizers. In turn, in the CloudChain prototype, each Stage is in charge of managing the orders (products included), which removes the requirement of a third party for message management, reducing the communication costs.

We analyzed the metric RT_{St} removing the costs of transportation of digital products from the evaluated prototypes to quantify the impact of resource management costs of both solutions and the costs of ensuring data through the transportation and storage layers. We compared the median ST_{St} of both solutions shown in Fig. 15 (vertical axis) for each value chain (horizontal axis). As it can be seen, in the CloudChain configuration a stable behavior is exhibited by stages in each Value Chain as the difference of median ST_{St} of all Stages for all chains is not significant. In turn, $MS_{Centralized}$ tends to increase the service time when adding services to the content delivery network because of the communication with the message manager.

CloudChain prototype is not affected by the number of Stages in a value chain, which means this scheme is suitable for building supply chains for the distribution of digital products.

7. Conclusion

In this paper, we have presented CloudChain, a new method for the distribution and delivery of digital products, which has been inspired by lean supply chain principles.

From a qualitative perspective, the deployment of CloudChain model in a prototype has shown the following benefits: (i) Flexibility: Stages can manage as many Catalog of digital products as alliances are established by the organizations and partners by creating value and supply chains. This enables organizations and partners to rapidly react to changes as Stages can be added/removed to/from a chain by creating/deleting Catalogs. Moreover, the administrators can determine the best cloud storage and content delivery networks suitable to manage the products of each Catalog in each value chain in which a Stage is included into. Multi-cloud and Content Delivery Service drivers

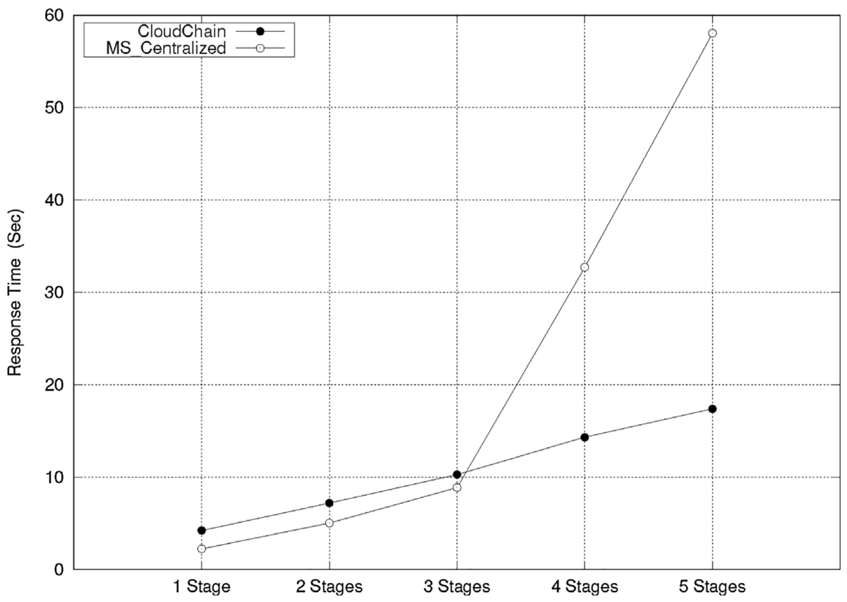

Fig. 14. The $RT_{VC/wf}$ produced by CloudChain and $MS_{Centralized}$ when increasing the number of stages in VC/workflows.

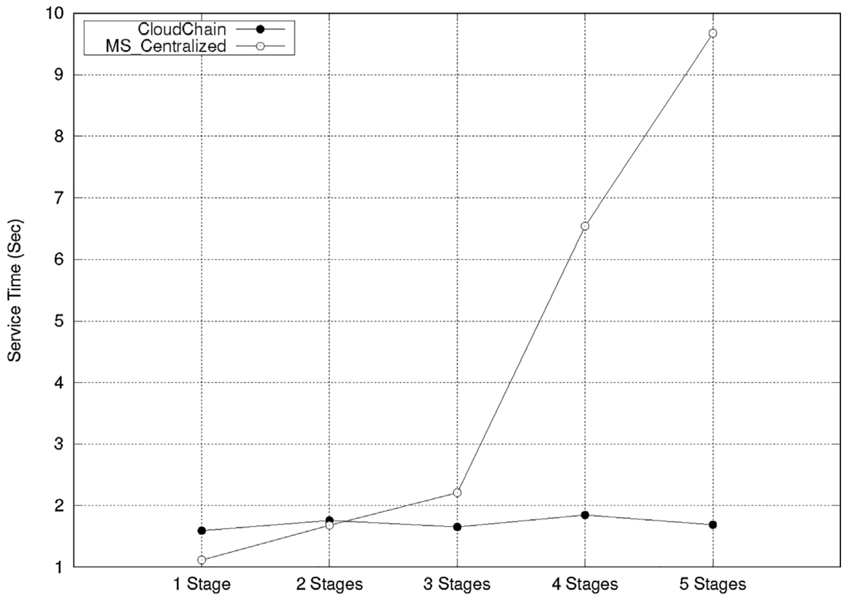

Fig. 15. The median service time of stages in value chains/workflows built with CloudChain and $MS_{Centralized}$.

are useful to manage transport and storage of products as it was shown in the evaluation of CloudChain prototype. (ii) Sovereignty: In CloudChain the negotiations to establish alliances are performed Stage to Stage; as a result, a Stage does not require knowing the deployment details of an entire value or supply chain to exchange digital products with other partners in a chain. (iii) Control and planning: The management supply chain architecture and the service of packing and logistics both enable partners to create seamless and continuous flow of digital products and metadata in a controlled manner. Non-functional components (SC-Tracker) enable users to follow up the life cycle of digital products and deliver information about Stages, chains, storage and transport performance, which improves the decision-making procedure. (iv) Security: The packing service can apply confidentiality, privacy and even integrity to the digital products by using a cryptographic scheme. This service can be exchangeable because of the modularity of the CloudChain architecture (SCM), which deploys security tools as black boxes invoked by packing service.

From a quantitative perspective, the evaluation of the CloudChain prototype (designed for organizational scenarios and deployed in the infrastructure of a private cloud) allowed characterizing the performance of main components of value and supply chains. The evaluation revealed worthy insights for decision makers about the impact on the Stage performance in sensing intervals to transport and manage digital products as well as the number of chains served by a Stage. Workload aspects such as the size of the digital products and concurrency issues were analyzed to suggest configurations of sensing interval for high and low concurrency workloads (number of concurrent value chains and digital products). In CloudChain, the size of the digital products does not affect the performance of the Stages when security is disabled, whereas the growth of service time in a Stage is proportional to the digital product size when security service is enabled.

The evaluation also included a comparison of CloudChain and a solution built with web services and message exchange synchronizers. The evaluation revealed that the performance of solutions based on coordination of messages is significantly affected by the costs of processing of messages and metadata when increasing the number of participants in a collaborative environment. In turn, the evaluation revealed that CloudChain increases the value chain size in the number of Stages without affecting the service time of the chained Stages as each Stage absorbs the costs of the management of digital products and the costs of metadata processing, which reduces significantly the communication overhead.

We are currently working on schemes for encapsulating applications into the Stages to build processing pipelines and fabrics for digital products and digital information goods controlled through supply chain management.

Acknowledgments

This work was partially funded by the sectorial fund of research, technological development and innovation in space activities of the Mexican National Council of Science and Technology (CONACYT) and the Mexican Space Agency (AEM), project No. 262891.

References

Amazon. Amazon simple storage service (amazon s3) (accessed 22.05.17).
Amazon Cloudfront, content delivery network (CDN) (accessed 22.05.17).
Anitha, Y., & Ravikiran, D. (2014). Revolution of storms from vendor lock-in to the data storage. *International Journal Computer Engineering in Research Trends, 1*(6), 391–396.
Beamon, B. M. (1998). Supply chain design and analysis: Models and methods. *International Journal of Production Economics, 55*(3), 281–294.
Berkhout, F., & Hertin, J. (2004). De-materialising and re-materialising: Digital technologies and the environment? *Futures, 36*(8), 903–920.
Bessani, A., Correia, M., Quaresma, B., André, F., & Sousa, P. (2011). *DEPSKY: Dependable and secure storage in a cloud-of-clouds. Proceedings of the sixth conference on computer systems, EuroSys '11*. New York, NY, USA: ACM31–46.
Bethencourt, J., Sahai, A., & Waters, B. (2007). *Ciphertext-policy attribute-based encryption. Proceedings of the 2007 IEEE symposium on security and privacy, SP '07*. Washington, DC, USA: IEEE Computer Society321–334.
Boneh, D. (2012). *Pairing-based cryptography: Past, present, and future*. Berlin, Heidelberg: Springer Berlin Heidelberg1.
Brodkin, J. (2014, January 13). *Dropbox messed up OS upgrade, caused two days of downtime*.
Buyya, R., Ranjan, R., & Calheiros, R. N. (2010). *InterCloud: Utility-oriented federation of cloud computing environments for scaling of application services. Proceedings of the 10th international conference on algorithms and architectures for parallel processing – Volume Part I, ICA3PP'10*. Berlin, Heidelberg: Springer-Verlag13–31.
Chow, R., Golle, P., Jakobsson, M., Shi, E., Staddon, J., Masuoka, R., et al. (2009). *Controlling data in the cloud: Outsourcing computation without outsourcing control. Proceedings of the 2009 ACM workshop on cloud computing security, CCSW '09*. New York, NY, USA: ACM85–90.
DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., et al. (2007). Dynamo: Amazon's highly available key-value store? *SIGOPS Operating Systems Review, 41*(6), 205–220.
DevOps (2017). https://devops.com/.
Emerson Network Power Independently conducted by Ponemon Institute LLC, D. C. P. B. S. (2016, January). Cost of data center outages.
Foster, I. (2005). *Globus toolkit version 4: Software for service-oriented systems. IFIP international conference on network and parallel computing*. Springer2–13.
Gonzalez, J., & Marcelin-Jimenez, R. (2011). Phoenix: A fault-tolerant distributed web storage based on URLS. *2011 IEEE 9th international symposium on parallel and distributed processing with applications (ISPA)*, 282–287.
Gonzalez, J., Perez, J. C., Sosa-Sosa, V. J., Sanchez, L. M., & Bergua, B. (2015). SkyCDS: A resilient content delivery service based on diversified cloud storage. *Simulation*

Modelling Practice and Theory, 54, 64–85.
Gonzalez, J. L., Sosa, V., Diaz-Perez, A., Carretero, J., & Marcelin-Jimenez, R. (2017). FedIDS: A federated cloud storage architecture and satellite image delivery service for building dependable geospatial platforms. *International Journal of Digital Earth*.
Gupta, P., Seetharaman, A., & Raj, J. R. (2013). The usage and adoption of cloud computing by small and medium businesses. *International Journal of Information Management, 33*(5), 861–874.
Havard Heitlo Holm, J.M.H., Gezer, V. Cloud flow project (accessed 22.05.17).
Holm, H. H., Hjelmervik, J. M., & Gezer, V. (2016). Cloudflow – An infrastructure for engineering workflows in the cloud. *The tenth international conference on mobile ubiquitous computing, systems (UBICOMM-16)*International Academy, Research, and Industry Association (IARIA), IARIA.
Jenkins Building, deploying and automating projects (accessed 22.05.17).
Josef Spillner, J. M., & Schill, A. (2013). Creating optimal cloud storage systems? *Future Generation Computer Systems, 4*(29), 1062–1072.
Josef Spillner, S. T., & Schill, A. (2014). *(mobiquitous '14). icst. NubiVis: A personal cloud file explorer*.
McMillan, R. (2017). *Amazon grapples with outage at AWS cloud service march 1, 2017 10:09 a.m.* .
Microsoft Azure storage (accessed 22.05.17).
Miller, F. P., Vandome, A. F., & McBrewster, J. (2009). *Advanced encryption standard.* Alpha Press.
Morales-Sandoval, M., & Diaz-Perez, A. (2015a). DET-ABE: A java API for data confidentiality and fine-grained access control from attribute based encryption. In N. R. Akram, & S. Jajodia (Eds.). *Information security theory and practice: 9th IFIP WG 11. 2 international conference, WISTP 2015* (pp. 104–119). Heraklion, Crete, Greece: Springer International Publishing.
Morales-Sandoval, M., & Diaz-Perez, A. (2015b). *DET-ABE: A java API for data confidentiality and fine-grained access control from attribute based encryption. IFIP international conference on information security theory and practice.* Springer104–119.
Morales-Sandoval, M., Gonzalez-Compean, J. L., Diaz-Perez, A., & Sosa-Sosa, V. J. (2017). A pairing-based cryptographic approach for data security in the cloud. *International Journal of Information Security,* 1–21.
Nasuni (2015). *Uncovers dangers of cloud storage provider lock-in*.
Nath, A., Saha, P., & Salehi-Sangari, E. (2008). Transforming supply chains in digital content delivery: A case study in apple. In L. Xu, A. Tjoa, & S. Chaudhry (Eds.). *Research and practical issues of enterprise information systems II, volume 255 of IFIP international federation for information processing* (pp. 1079–1089). Springer US.
OnApp The hybrid cloud management platform (accessed 22.05.17).
Opara-Martins, J., Sahandi, R., & Tian, F. (2014). Critical review of vendor lock-in and its impact on adoption of cloud computing. *2014 international conference on in information society (i-Society),* 92–97.
OpenStack (2015). OpenStack.
Porter, C. C. (2012, January 20). *De-identified data and third party data mining: The risk of re-identification of personal information*.
RabbitMQ RabbitMQ messaging (accessed 22.05.17).
Rosenberg, B. (2010). *Handbook of financial cryptography and security* (1st ed.). Chapman & Hall/CRC.
Salveggio, E. (2004). *Your (un)reasonable expectations for privacy. Ubiquity*. ACM.
Sloan, R. H., & Warner, R. (2013). *Unauthorized access: The crisis in online privacy and security.* CRC Press first edit edition.
Stark, J. (2015). *Product lifecycle management. Product lifecycle management (Volume 1).* Springer1–29.
Team, M. E. (2014). *Microsoft solutions for managing the media supply chain: A services oriented architecture blueprint for the agile media company*.
Toosi, A. N., Calheiros, R. N., & Buyya, R. (2014). Interconnected cloud computing environments: Challenges, taxonomy, and survey. *ACM Computing Surveys, 47*(1), 7:1–7:47.
Williams, A. (2013). *It's official, the Nirvanix cloud storage service is shutting down*.
Xiong, H., Zhang, X., Zhu, W., & Yao, D. (2012). CloudSeal: End-to-end content protection in cloud-based storage and delivery services. *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, 96*, 491–500 LNICST.
Yanez-Sierra, J., Diaz-Perez, A., Sosa-Sosa, V., & Gonzalez, J. L. (2015). Towards secure and dependable cloud storage based on user-defined workflows. *2015 IEEE 2nd international conference on cyber security and cloud computing (CSCloud),* 405–410.
Yun, W., Cegielski, C. G., Hazen, B. T., & Hall, D. J. (2013). *Cloud computing in support of supply chain information system infrastructure: Understanding when to go to the cloud*.
Zhou, H., & Benton, W. C., Jr. (2007). Supply chain practice and information sharing. *Journal of Operations Management, 25*(6), 1348–1365 Supply Chain Management in a Sustainable EnvironmentSpecial Issue on Frontiers of Empirical Supply Chain Research.

Gerardo A. Vazquez-Martinez received a Master degree in information technologies from Center of Research and Advanced Studies of the National Polytechnic Institute, (CINVESTAV), Ciudad Victoria, Mexico. His expertise areas are cloud computing, content delivery and storage virtualization.

J.L. Gonzalez-Compean received his PhD in Computer architecture from UPC Universitat Politécnica de Catalunya, Barcelona (2009). He was a visiting professor at Universidad Carlos III de Madrid, Spain (2011) and now he is full time professor researcher at Center of Research and Advanced Studies of the National Polytechnic Institute, (CINVESTAV), Ciudad Victoria, Mexico. His research lines are Cloud computing and containerized Storage systems, linguistic archival systems, federated storage networks. His expertise areas are the design of fault-tolerant, adaptability and availability strategies as well as task scheduling and storage virtualization.

Victor J. Sosa-Sosa is a full-time research-professor CINVESTAV – Tamaulipas research center. Currently, he was a visiting professor in the Database and Information System group at Max Planck Institute für Informatik in Germany. He has a PhD in Computer Science from Technical University of Catalonia (UPC-Barcelona). Participant and responsible of research and development projects funded by the National Council for Science and Technology of Mexico (CONACYT) and private companies. He has been advisor of more than 15 master's thesis and 2 PhD thesis. His research interest and specialization areas are related to information management, Distributed Information Systems and Databases, distributed information at Web scale, especially on topics related to efficient information search, storage and analysis, integration of Web query interfaces (DeepWeb) and knowledge harvesting.

Miguel Morales-Sandoval received PhD degree in Computer Science in 2008 from the National Institute for Astrophysics, Optics and Electronics (INAOE), in the Computer Science Department. Postdoctoral fellow in the Center for Research and Advanced Studies of the National Polytechnic Institute (Cinvestav) campus Tamaulipas from 2012 to 2014. He was a visiting professor from 2014 to 2016 at Cinvestav and now he is a full time professor-researcher (Cinvestav-3B level) in Cinvestav Tamaulipas. His research areas are cryptographic algorithms and protocols for embedded systems, data security in constrained devices and in the cloud, embedded system design with HDLs and FPGAs, reconfigurable computing as well as software and hardware engineering.

Jesus Carretero Perez is a senior full time professor-researcher at Carlos III University, Computer Science and Engineering department, Madrid, and his major research lines are high-performance computing and cloud computing, parallel and distributed systems, computational linguistics and real-time systems. He leader of research group ARCOS at Carlos III University and he is also the director of the Master in Administration and Management of Information Systems and has published 17 books, has been involved in 52 research projects and he has written 200 journal and congress articles.