ISSN 1507-3858

Michael C. Jaeger
Siemens AG, Corporate Technology, Germany
michael.c.jaeger@siemens.com

Alberto Messina
Center for Research and Technological Innovation, Italy
a.messina@rai.it

Mirko Lorenz
Deutsche Welle, Germany
mirko.lorenz@googlemail.com

Spyridon V. Gogouvitis, Dimosthenis Kyriazis
National Technical University of Athens, Greece
{spyrosg, dkyr}@telecom.ntua.gr

Elliot K. Kolodner
IBM Haifa Research Lab, Israel
kolodner@il.ibm.com

Xiaomeng Su
Telenor R&I, Norway
xiaomeng.su@telenor.com

Enver Bahar
Siemens AG, Corporate Technology, Germany
enver.bahar@siemens.com

CONTENT-CENTRIC STORAGE: ACCESSING DATA OBJECTS BASED ON METADATA AND RELATIONSHIPS[1]

Abstract: Emerging storage cloud environments aim at facilitating the storage, retrieval, association, analysis and distribution of data and content with one of the fastest growing content types being fixed content (also known as object storage). Fixed content often has rich, structured and mutable metadata, which enables finding, managing and extracting value from the content as well as providing information regarding the meaning of the object and its relationship to other objects. In this paper we introduce an approach, namely content-centric storage, which allows applications to access data objects through information about their content

[1] Selected parts of this article were published under non-exclusive copyright in Proceedings of the Federated Conference on Computer Science and Information Systems FedCSIS 2012 (see [Jaeger et al. 2012]).

rather than their path in a hierarchical structure. The applications do not need any knowledge about the data store organization or the place in a storage hierarchy. Rather, the applications may query for the desired content based on metadata associated with the data objects. We also demonstrate the operation of the implemented approach and evaluate its effectiveness using a real-world scenario, based on a media production application.

Keywords: cloud storage, content-centric, metadata, data modeling, object storage, distributed systems, server applications.

1. Introduction

Cloud environments are facing a new challenge: the explosion of personal and organizational digital data. According to an IDC study [IDC 2010], the information will grow by a factor of 44 reaching 35ZB in 2020 (while in 2009 the corresponding figure was 0.9ZB) and the entities holding this information (referring to files and containers) will grow by a factor of 67. The latter highlights the fact that in the emerging era of the Future Internet, the explosion of raw data and the dependence on data services is expected to be further amplified due to the strong proliferation of data-intensive services. Apart from the challenge of number and size, there is also a challenge of distribution, where users expect to have the data available anytime, anywhere and through any device. All these applications require powerful storage systems capable of handling large files and/or a large number of files. Handling refers to the ability to easily ingest content of any type to efficiently obtain the desired content and smoothly access the content through desired devices. These functionalities go beyond plain data storage functionality as offered by cloud storage providers today.

The picture nowadays regarding access to storage is as follows: data files (organized by a hierarchy or structure) are used as a placeholder of content and users navigate through this structure when searching for particular content. However, if the amount of data is huge, it becomes ever more difficult to maintain such a hierarchy and correctly allocate a file so that one can easily find it later. In the proposed approach to content-centric storage, the user is not limited to using hierarchies to find his or her content. Rather, either the user or the system describes the content through metadata and then accesses the content based on its associated metadata. This approach to content-centric storage requires efficient software to automatically create or ingest, and then later retrieve, any kind of metadata about the content. To further illustrate our intention and scope, we consider an example from YouTube. Each video on YouTube can have a set of metadata associated with it. The metadata might have different characteristics. Some are static (e.g. “title”), others are dynamic (e.g. “tags” or “most viewed”); some are content-related (e.g. “category”) and others are technical (e.g. “length of the video”). With YouTube, one can access the video by searching metadata. For example, a user can ask for a video that is “most viewed”

or that belongs to the "category" drama. Our goal is to enable the same function, but generalize it and extend its scope to any type of data, not just video. We can see that such metadata becomes ever more valuable when more and more areas of life are touched by all types of data – text, pictures and moving images – with updates from sensors, mobile clients, etc.

Furthermore, one needs to consider that data and content are quite dynamic. Their properties and interdependencies often change resulting in the corresponding changes in their storage location (e.g. due to migration or replication of content), representation and access paths. Content-centric storage aims at enabling users to obtain such dynamic content by ensuring that the aforementioned implied changes are transparent to them. Besides this enhanced access to data objects, the approach introduced in this paper allows for the development of content networks. Sets of data objects are in many cases used within an application or requested by the users, and thus a challenge that arises in storage clouds refers to the identification of the relationships between objects in order to optimize both the management of these objects and the underlying infrastructure as well as the service provisioning. Through the rich metadata attached to data objects, content-centric storage enables the expression of relationships between different objects and as a result the development and utilization of the corresponding content networks.

The content-centric storage approach introduced in this paper is being developed in the framework of an EU-funded research project, namely VISION Cloud [Kolodner et al. 2011; Kolodner et al. 2012], which aims at developing a cloud storage environment that allows for the provision of data intensive storage services. Unlike Amazon S3 [Amazon Simple…], Microsoft Blob Service [Microsoft Azure…] or hardware appliances, VISION Cloud also supports private cloud installations and does not require the use of specialized hardware. In addition to content-centric storage, VISION Cloud [VISION Cloud…] introduces more innovations (e.g. computational storage, data mobility, enhanced security and compliance, etc.) for cloud storage systems.

In this work, we present in detail the motivation, the requirements and the design for a metadata-enhanced large storage system, which we call content-centric storage. This article is structured as follows: we present the motivating scenario in Section 2 and derive the corresponding requirements in Section 3. Section 4 introduces the particular effort resulting from requirements that come from different domains and explains our solution for such a setup. This section also includes results from performance tests. Finally, the related work is presented in Section 5 and brief conclusions are given in Section 6.

2. Illustration of concepts

It is useful to look first at similar solutions from specific domains. Popular Internet applications already provide examples for the content-centric approach to accessing data, e.g. the aforementioned YouTube site. This site does not just provide documents,

it also provides them on a large scale. Another example is the Amazon online store that demonstrates some of the anticipated functionality and some of the application requirements for content-centric storage. Amazon offers the user access to a product database that contains the items that are (or were) offered. An Amazon user starts with a search for a product, which is similar to a search for content in an object store for documents. Figure 1 outlines the various elements that Amazon provides to the user in order to support a search for the appropriate product. In many cases, the means provided are very parallel to content-centric functions that should be provided by an object store:

1. **Keyword search.** This search is optimized for returning the relevant results. A single name or title may not be sufficient, because a title does not necessarily identify the content of the object. Instead, the user can search for a combination of descriptions or tags that have been assigned to an object in order to indicate its relevance to particular topics. When Amazon returns the results of a query, it also shows additional keywords associated with the result items that were not part of the query. This helps the user to refine the search with more appropriate search terms.

2. **Popularity.** A rating by other users is provided as an estimation of the relevance of the search results. Other users can rate a product or content in order to support its selection for subsequent searches. In addition to content-centric storage, VISION Cloud introduces more innovations (e.g. computational storage) for cloud storage systems. More information can be found on the project's website [VISION Cloud…].

3. **Cost.** The price for the item is shown in Amazon. For data objects, a similar metric-like price is relevant as well – the cost of access. Thus, the price for an item can be seen as a metaphor for the effort to access the object in the storage.

4. **Logistics.** The availability is shown. For data objects this refers to the transfer duration, which is relevant for large objects (e.g. video). In particular, the transfer duration would depend on the location of the object and its size. This could allow applications to optimize their downloads of data objects.

5. **Preview.** For data objects that allow for a meaningful preview function (e.g. 3D images, video) this enhances the user's selection of results. It could also improve the efficiency of the system, since the number of download requests for a closer examination of a data object is reduced.

6. **Mining bought items.** Amazon reports on the items that were bought in combination with the current item. For data objects, this could refer to other objects downloaded together with the current one during past user sessions. A single incidence may not provide meaningful recommendations; but taken over a very large number of sessions this information could be relevant.

7. **Mining searched items.** Amazon reports items that were searched for together with the current one. For data objects, this could refer to other objects returned by searches together with the current one during a user session. Again, a single incidence may not provide meaningful recommendations, but taken over a very large number of sessions this could lead to meaningful information.

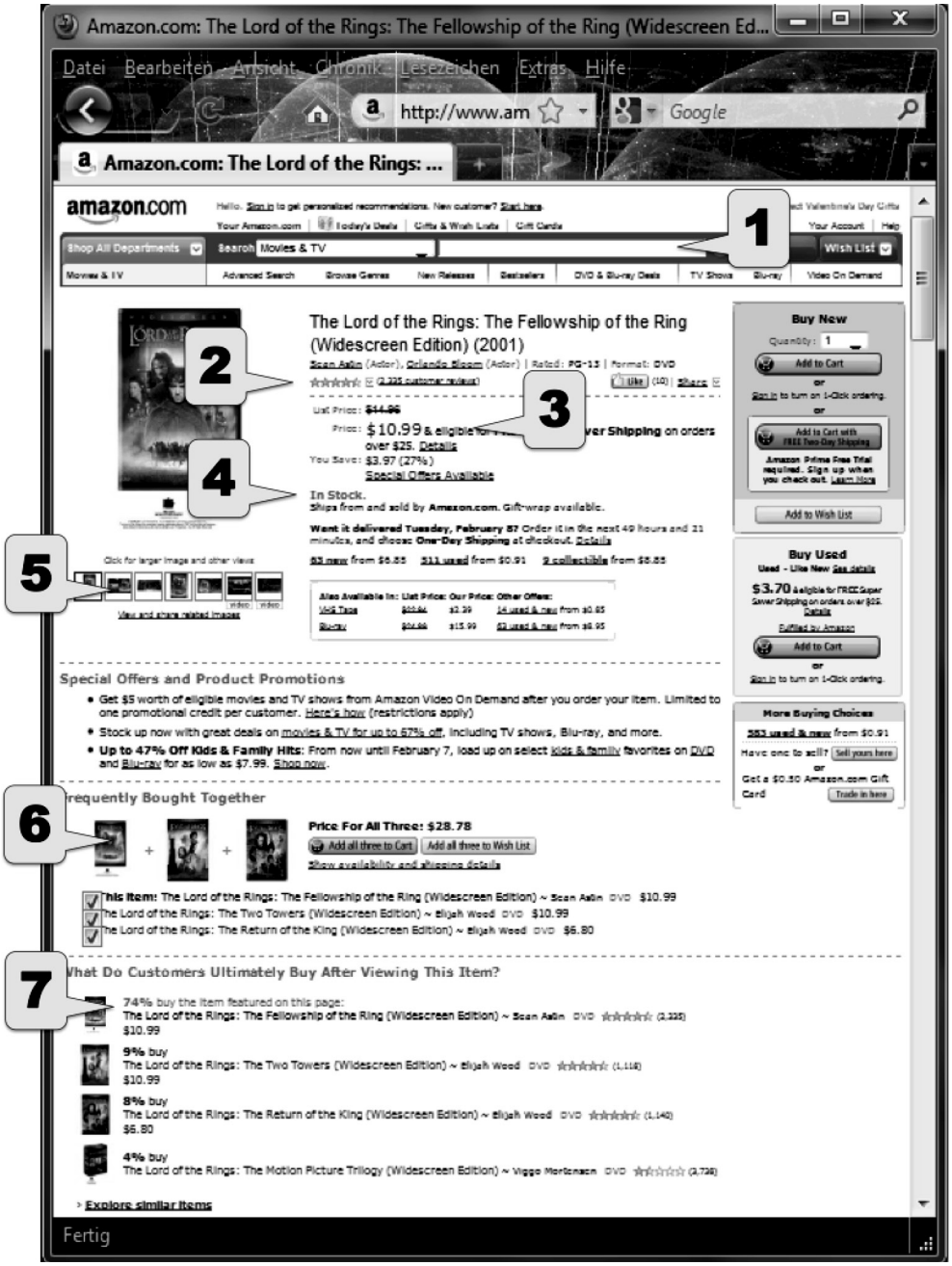

Figure 1. An example of the idea of the content-centric approach: relevant information for picking an item at Amazon.com

Source: own elaboration.

All these items provide representative information about the product. The user does not see the product itself. However, the information presented about the products helps the user to choose the right item. Usually, a user will find the right product based on the metadata. In rare cases, a user will navigate through a hierarchy of goods to directly identify the right product by its title. The YouTube example from the introduction presents a similar approach. As with Amazon, the user usually does not navigate through a hierarchy to identify a particular video file. Rather, the user searches, selects, browses, looks at previews and considers the number of clicks to ultimately find a video that suits his or her interests. The aim of the content-centric storage is to enable such functionality for any kind of data, not limited or dedicated to a product store or a video platform with the following benefits:

a) Improving effectiveness. The content-centric interface enables an application accessing storage, and therefore also the user of the application to find the desired data objects more effectively.

b) Improving efficiency. In addition, users/application can access content more efficiently, because the storage can optimize the way it stores data internally and the way it retrieves data based on metadata.

Regarding the example given through the Amazon case, most of the information required to find an object will be provided through the metadata associated with it. Often this metadata can be extracted automatically from the object content. In addition, the storage system or the application can collect operational information, such as the combination of accesses, the tags that are searched, etc. In other cases, users provide additional information, such as a rating and the application stores such descriptive data as metadata associated with an object in the storage.

3. Requirements

Within the context of the VISION Cloud project [VISION Cloud…], a requirements gathering process has been executed based on four distinctive use cases from different domains. More precisely, the areas of telecommunications, media, health care and enterprise have been examined to elicit requirements for content-centric storage. Here, we will mainly focus on the media domain, but the processes and concepts described can be extended to various different scenarios and use cases.

The efficient access to stored data is of major importance to a media application. Therefore, the content-centric storage API shall allow the user to access the data objects through the specification of content-related features and not limit him or her to the traditional location based approach of hierarchical file systems. Therefore, an application should be able to query for data objects by information about their content, which is expressed as metadata elements. Metadata can be multi-faceted and extremely useful:

a) Metadata about content. Metadata can include information describing the content and informing the user of what he or she will see, read or hear when accessing

the data object. This can be different depending on the data object. For example, a text document would include a title, while a video object would include the tagging of each scene.

b) New filters for content can be created based on metadata. For example, metadata can be used to provide a measure of popularity for a video, as can be achieved by combining metadata fields, such as "likes on Facebook", "most commented" and "most viewed". This can provide new insight into the content itself.

c) Proximity information according to a metric. For a given metric regarding data object contents, the calculated proximity could be of interest for a range of applications. For example, a location-based search given a 2D metric system on spatial coordinates can be constructed based on appropriate metadata fields.

d) Relations between objects can be expressed. For example, metadata can be used to indicate that an object belongs to an ordered or unordered set and queries can be executed based on these relations. A producer, searching in the archive of media assets, requires information about which audio tracks belong to which video tracks, or which material belongs together because it represents a sequence of shots, etc.

For efficient media production, some features are desired by the production environment/application, which in turn pose requirements to the underlying content-centric storage.

The ingestion of metadata is of major importance in the media domain. The process should be automated and simplified allowing for richer metadata to be added to the data objects. Table 1 shows metadata fields of objects as stored in YouTube, Flickr and Twitter. Accessing and parsing these fields is a straightforward process providing the basis for a very powerful approach to leveraging existing metadata. A more complex algorithm could be used that is able to use specific indicators (likes, views, dislikes, number of comments, rating) or combinations of them to find the videos, photos or tweets that are the most important, filtering out the rest. Extending the approach, ingesting data from other platforms in this way enables varied sources without losing the metadata which is the foundation for innovative content-centric access.

Table 1. Overview of metadata available from different sources

YouTube		Flickr	Twitter	
Channel	Views	Public Contact List	ID	language
Video ID	Comments	Public Favorite List	Name	verified
Title	Likes	List of galleries to which a photo has	Screen_name	geo_enabled
Description	Dislikes	been added	location	followers_count
Duration	Rating	List of the photos in a gallery	description	favorites_count
Category	Favorites	List of public photos for a user	profile_image_url	friends_count
Tags	Embeds	Standard infos (sizes, tags describing what is shown)	url	statuses_count

Source: own elaboration.

Existing metadata can also be used to express relationships among the objects. Such structural information is an important type of metadata in many professional and

consumer scenarios. A typical example would be craft editing, which is usually done externally, through the use of dedicated software. When exporting the material to the editing workstation, it is of vital importance to preserve the structural relationships among the objects. For example, the fact that the tracks are part of the materials or that shots and effects are part of the video track need to be preserved in all cases and enable the maximization of both the storage efficiency as well as the quality of the production.

Generally, based on their intrinsic properties, we can identify a classification of relations that can be used to make access to the stored objects more efficient, e.g. by using pre-fetching and intelligent caching mechanisms. These properties of relations are order, transitivity, equivalence and subsumption. We provide some examples of the usefulness of each:

a) In media production it is required to represent the ordered relation between a video track of a piece of material and the individual frames that form the video content in a specified and unambiguous sequence, being both the individual frames and the track objects uploaded onto the storage cloud.

b) As for the transitive relations, in media production it is required to allow queries that retrieve all the parts of a specific object regardless of the level of the partition hierarchy in which they are located (a part may be itself partitioned into sub-parts in a recursive way).

c) Equivalence relations are used to partition sets into subsets, which contain equivalent objects under a specified criterion. Furthermore, it may be required for example to be able to see different encodings of the same piece of material as an equivalent from the point of view of the expressed content. This feature is also relevant for content delivery when different resolutions or encodings of the same video material are transferred and displayed to the end-consumer according to the different device that he or she uses.

d) Subsumption relations are required when there is inheritance between classes of objects, and there is a requirement to manage objects belonging to a specified superclass regardless of their specific subclass. For example, visual shots (the abstract superclass) can include at least the following subclasses: hard cuts, dissolves, wipes.

Though in practice many (potentially unlimited) relations among objects are possible, these relations can be classified according to set-theoretical approaches and limited to simple relations, such as equivalence, order, transitivity, and subsumption, which are important across multiple application domains. Thus, content-centric storage must support these relations.

4. Content quality

The relevancy of metadata is obvious for identifying, characterizing and managing data objects. However, one problem with metadata is actually the quality of the provided information. The quality of the provided information is the relevant factor

for how well a data object can be found and how popular a data object will be. We name the quality if the metadata has content quality. Content quality, referring to the data object's metadata, represents a relevant issue for the advanced handling of data objects.

Caring for the content quality allows applications, and therefore users, to distinguish good content from "not-so-good" content. Therefore, it enables browsing the content more efficiently. Assuming that "good" content will be requested more often, it could help the storage to optimize the content delivery according to this implied popularity or the creation of replicas. Accordingly, other areas apply in a similar manner, for example, compliance issues in connection with the replica creation, etc. Also, dedicated metadata tags may represent quality attributes, which are processed by the operating layer of the VISION Cloud system. Such information can help an object storage system to manage data more efficiently and serve requests of applications more effectively.

Accordingly, other scenarios, where metadata has an impact on storage management, are valid in a similar manner. For example, metadata can have compliance information that cover locality or confidentiality issues in connection with the replica creation, etc.

4.1. Content quality from the application perspective

If we think about an archive for media files, either if the scenario targets the production of broadcast material or we have many cell phone subscribers that share millions of video clips, content quality provides the ability to identify the right media, the popular video clips, the most viewed videos.

This aspect concerns the following questions. How quickly and easily is the right piece of media found? For example, will the user find only poor material and waste hours browsing the storage or will the user go from one interesting media piece to another? Thinking of the media production scenario, the second question is: how quickly can a user or an editor clarify the usage conditions, with regard to technical or legal constraints? For example, can the system actually process information that covers legal constraints in order to ease the handling with different usage licenses? It is all about saving time and focus on the actual job, which is producing media. How quickly can a user or an editor take a media file and then proceed with the actual editing job? Or how quickly can the user find interesting video clips and continue with the next interesting one without being frustrated?

4.2. Degree of how much metadata

Metadata should be comprehensive and complete for all. The measure of completeness refers to the formats of origin or formats that are relevant for the media domain. Contrary to older approaches or the reality in everyday work, it is not about parts

of the original metadata or the so-called "three tags"? For example, if the origin of metadata is YouTube, it is considered good quality if most of the YouTube formats are actually supported, and accordingly, the available metadata from YouTube is taken for further processing. There are already some approaches for measuring the degree of metadata, for example from the knowledge management domain.

Conclusion 1: The degree of completeness of metadata is a quality criterion.

4.3. Changing metadata

Over time, the value and interpretation of content will change. Therefore, metadata should evolve. This means that if a content item or even a large repository of content is used over time, the metadata should meaningfully reflect the patterns of that usage, e.g. what was the last modification of the metadata catalogue? This concept could even be used to safely delete certain content or at least move it into a long-term storage archive, thus freeing an active system under load.

The key idea is that if a file that sits for years in the storage without being touched (number of uses, number of downloads or number of views being the metadata here), then this file is presumably of less value than others. With such novel ways of machine-based differentiation of content, the problem of constantly growing storage of all data and all files could be managed in the future.

Conclusion 2: Longevity of the metadata is a quality criterion.

4.4. Traveling data

Another characteristic that does not directly express the quality of the content is the anticipated mobility of the data in terms of projected use. Such mobility has implications on the quality of service. Imagine that some of the telecommunications customers were often on the move. Suppose, for example, that a customer from Telenor often traveled between Norway and Pakistan. He might benefit from accessing his or her work-related documents in the right country at the right time. From a customer perspective, this has some implications, for example to save data roaming cost if access through cell phone or mobile Internet access plan takes place. Moreover, if data can migrate along with the user, the user's experience will be better.

Regarding the Pakistani/Norwegian customer that travels regularly between Norway and Pakistan: the associated data could follow the moves of this customer in order to improve access latency to the customer's data. One possibility could be that the customer needs to fill one metadata field to tell the system that he or she wants the data available in Norway and Pakistan in a given period of time. Then, VISION Cloud Norway and VISION Cloud Pakistan can keep a synchronized copy of the documents at both locations.

It must be considered when talking about "quality attributes" that they should not be there only for this purpose and be modified by the user to trigger some behavior of the system. Instead, they could be represented by existing metadata that happens to be useful to optimize the behavior of the storage. As an alternative, it is at least some metadata that is updated by the application or a storlet or the system itself.

Referring to the Pakistan-Norway example, another technical solution without requiring the interaction of the user is to have an "Access location" metadata (or set of it) that would store from where an object (or a container holding the relevant objects) is accessed. There could be a counter showing how many times an object (or container) has been accessed in a zone (country or smaller). Over time, the system would migrate (or replicate) the objects to a data center close to the most used zone.

A third technical proposal is using mobility information. This implies having the current location of a user (GPS coordinates, access network, cell, etc.) stored as metadata (this involves having users stored as objects). Depending on the current location of a user and the objects that he or she is accessing, the system could anticipate the migration or replication of objects in the same container (or having some similar metadata).

Currently it is not expected to pre-present users as data objects in VISION Cloud, such information should be managed by means of case application. A possible technical implementation could be the "quality attribute" management of VISION Cloud, which can use the LDAP attributes of users that should be available for them.

Conclusion 3: Metadata expresses the access location in future use.

4.5. Using data

Especially for the media domain, the presence of copyright and user licenses is very important for the proposed use of the media and even more, preventing the organization from paying penalty fees. The technical relation with the storage system lies in the support for access policies based on such information. Basic information is comprised of the following items: a) complete rights information; b) what is the license? c) what are the license conditions? d) source information; e) who has created this content? and f) who has ingested that content?

Conclusion 4: For media-processing domains, the degree of open or closed licensing can also be seen as a quality measure.

4.6. Placement and locality

We consider an example from a telecommunications provider. By law, banks in Norway need to record all the conversations that their financial advisors have with their customers. We assume that a storage cloud is used for this purpose. It is likely

that such data cannot be freely moved out of Norway. The basic issue is that a state prosecutor must be able to confiscate the storage if a criminal action prosecution needs to take place.

Likewise, some personal information cannot be moved out of the EU, and some information from one country simply cannot be moved to another (Telenor Bangladesh and Telenor Pakistan, for instance). The technical implication here is that there should be metadata that tells the storage system the constraints of moving or copying data from one place to another (e.g. the data protection regulation that has to be complied with). The case of healthcare use has similar concerns.

Conclusion 5: This is a classic compliance/placement case as this scenario describes compliance with existing data protection laws.

4.7. Summary

The discussion has yielded a summary of data quality relevant characteristics that could be maintained by VISION Cloud storage that serves as the content quality. This input targets primarily the discussion of advanced characteristics for VISION Cloud storage. In summary, the points were:

– completeness of metadata,
– liveness of metadata,
– license and Usage Information,
– anticipated location of access,
– locality restrictions according to compliance issues (however, it must be noted that this issue does not represent a sole "Content Quality" topic, but is also covered by the compliance issues with placement of data from a legal point of view).

The application use cases have described these points according to their domain characteristics. Subsequent work needs to consider this input for a domain-independent mechanism of advanced storage management using the above explained quality attributes.

5. Design and implementation

Our content-centric approach is implemented by using a document store in conjunction with an access service component placed above it. The access component is comprised of several modules based on our analysis of the VISION Cloud project use cases. Each module offers a self-contained set of functionality. Figure 2 summarizes the modules of the interface:

a) A basic metadata file service allows putting new data objects and getting existing data objects and its metadata, similar to conventional document storage solutions, such as Couch DB. This module wraps the interface of the underlying object service, which provides basic operations on objects and their metadata.

b) The relations service provides various ways to handle objects based on their metadata. This includes querying for objects by metadata keys and values. Apart from tagging, it also supports relations over objects through the controlled setting of metadata. For example, an application can indicate that certain data objects belong to a set. Accordingly, an application can also query objects belonging to a set.

c) A metrics service allows setting metric values for particular data objects. Accordingly, data objects can be queried according to a metric statement. This functionality is useful, for example, to attach geographical (two-dimensional) coordinates to an object. Then, a query can identify data objects that are in the vicinity of a defined geographic coordinate.

d) The module for operational statistics allows access to information of statistics maintained by the system. Such statistics include the number of downloads, or the number of accesses during queries or the date and time of the last download.

e) Another module is for the import of metadata from existing sources, which is explained in the next section.

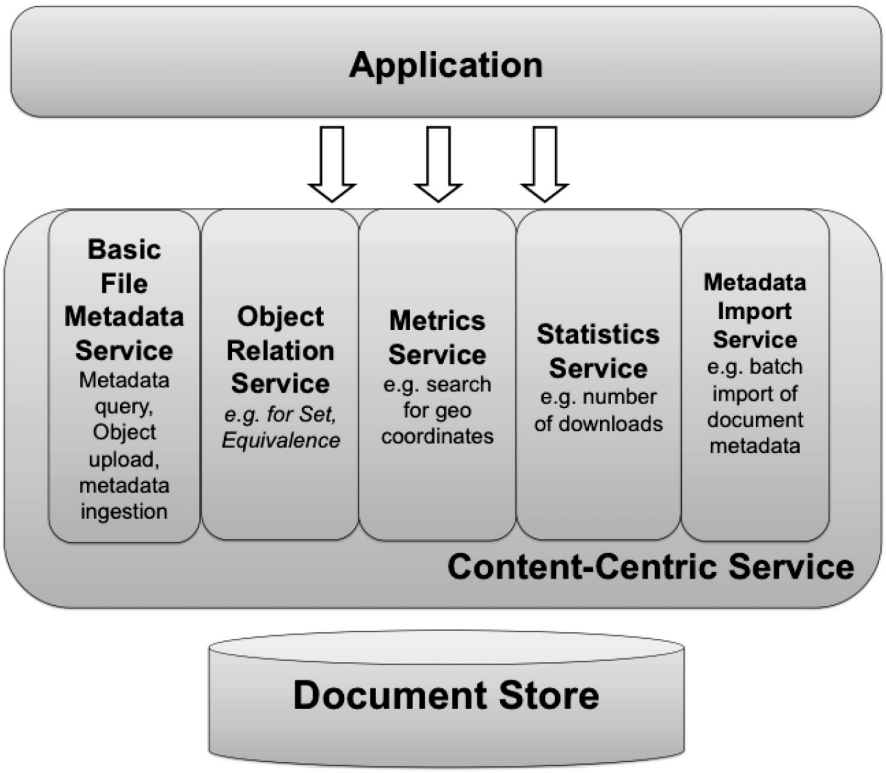

Figure 2. Content-centric service

Source: own elaboration.

5.1. Metadata import

The key element for the metadata approach is the re-use of existing metadata. Therefore, the content-centric API accepts uploads of metadata documents and associates this data with the data objects in the store. For the upload, the metadata is expressed in an XML document. In the considered domains, such an XML document can be easily extracted from the used file formats. The implemented approach works as follows:

a) A person creates an XML Schema document, which defines the format of the metadata to be associated with key-value pairs of the data objects.

b) The definition of an XML Schema also defines how to process information about relations between data objects using special XML tag attributes.

c) The client uploads this schema to the data store, technically as another plain data object to the store.

d) The client assigns an identification for the schema document to the uploaded XML schema data object in order to refer to this identification at later metadata imports.

e) After the initial upload of the schema, the client can upload XML documents containing the metadata conforming to a specific schema. The upload request of the XML metadata includes the XML schema reference.

This functionality enables the re-use of existing metadata. Different sources are possible, for example, the import of YouTube metadata associated with a YouTube video. Other representations than XML require conversion to XML. Existing XML documents containing object metadata can be annotated and then uploaded in larger batches for import into the storage.

5.2. Architecture

VISION Cloud employs a catalog implementing an existing document database, called NoSQL, to hold the metadata associated with data objects and through queries on the metadata to access objects according to their metadata values. The NoSQL-based approach was chosen to achieve a high level of scalability. The key-value data model is not sufficient to support efficiently the processing of all semantic information regarding content. Accordingly, the semantic information is stored in the metadata in the NoSQL key-value model, but some of its processing is done by the content-centric modules running above it.

Architectural challenges also arise from handling semantic information since VISION Cloud is designed to be a highly distributed system with high performance, scalability and availability. Thus, the CAP theorem [Brewer 2000] also applies to a global metadata schema or semantic information by the application if this information is subject to changes during the use of the system. Thus, we can choose to maintain for the used metadata, according to relations, for example, any two out of the three properties:

– consistency,
– availability and
– partition tolerance.

Overall, for a storage that implements a large base of data that must be distributed, we have chosen availability and partition tolerance as being the most important, so following this decision for a metadata schema for information by the application, the application using VISION Cloud must be able to deal with eventual consistency.

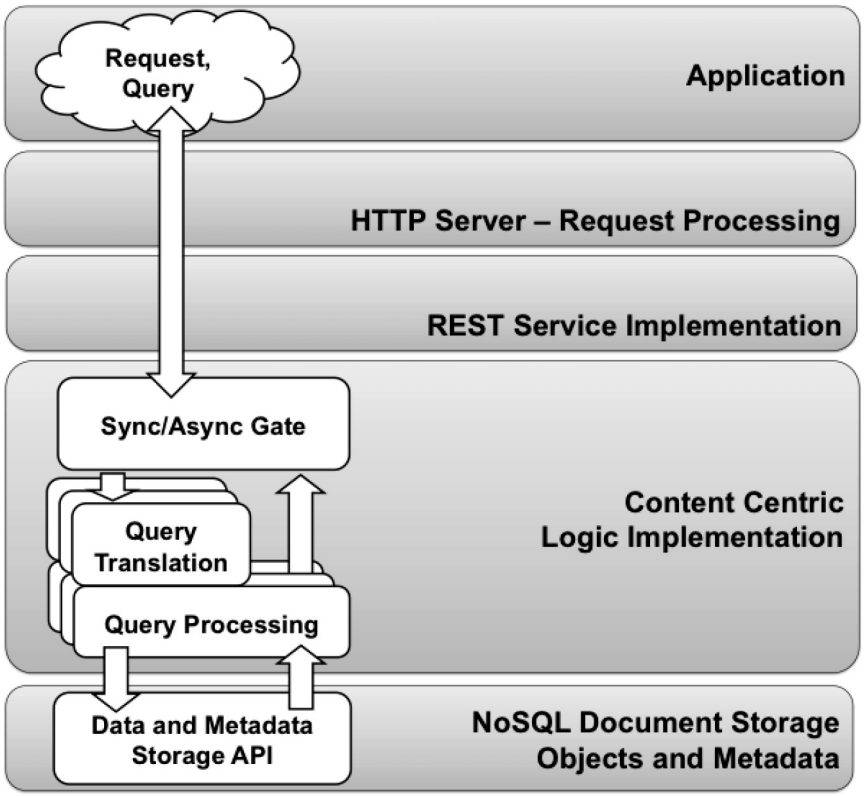

Figure 3. Content-centric service implementation: runtime view

Source: own elaboration.

5.3. Runtime view

The design for the content-centric modules is critical as it directly impacts the performance of applications working with the storage. The demands for this access are performance, scalability and the use of a parallelized infrastructure (for example, the object service, a distributed object store of VISION Cloud [Kolodner et al.

2011]). The modules presented above implement the cloud computing paradigm of stateless request handling using two main stages: a query translation stage and a query issuing stage. When an application issues a query to the content-centric interface, the content-centric storage processing part translates this into one or more queries on the underlying (NoSQL) document storage.

Therefore, the content-centric modules translate queries posed via the rich query API into basic metadata queries for the key-value pairs that are supported internally by the metadata catalog. Regarding a cloud-paradigm for the design of applications, the processing of this translation and the re-query operations are designed in an asynchronous and parallelizable way. Figure 3 outlines the general approach. Queries are received from the application in a synchronous manner, while internally queries are issued asynchronously to the internal storage subsystem.

An example would be as follows: the client software submits a query to the content-centric API, expressing several key-value statements combined with a logical AND. The rest arrives at the REST service implementation where request parameters are decoded. An implementation layer parses this query and creates several individual queries to the underlying storage. This storage allows only querying for one key-value pair at the same time. When all the results from the individual queries are received, the intersection from the results is generated and returned to the client.

5.4. Implementation

Figures 2 and 3 also imply a particular type of interface for the content-centric storage – it offers a HTTP-based RESTful service interface to the applications, allowing access to data objects from a resource point of view. Accordingly, the old vision of a service component offering a set of operations is replaced by a set of resources. A resource can be a data object or a set relation among data objects that is created (using a PUT-http-request), updated (POST), queried (GET) or deleted (DELETE). Technically, a request to a resource is then transformed into one or more queries to the underlying storage. The layer where the content-centric functions are implemented serves as a middle tier providing a rich interface to the application and using the objects storage, which involves the NoSQL document database.

Since the basic functionality of the content-centric access is based on the definition of relations among the data objects (cf. Section III), the consequent design of a RESTful service results is that each relation between data objects turns into a resource as well. Thus, for example to denote a set of objects by using the set relation, the client application can use the RESTful service interface to ”PUT”, i.e. to create a relation, to add data objects to the set-relation by issuing a POST-request, to query the relations using GET-requests. A particular issue with this approach is the consistent design of the resource paths with the RESTful storage interface of the underlying object storage.

5.5. Performance tests

We have conducted a set of performance tests in order to evaluate the query performance of the content-centric service implementation on a single computer or network node. The deployment involved the content-centric service as a Java servlet implementation running in a Tomcat 7 servlet container (exact version 7.0.26). The REST services were implemented using the Jersey Javax-rs framework. The service accessed an underlying NoSQL document storage as a persistence layer. For the tests, the CouchDB version 1.1.0 database server has been used.

Three test platforms were set up: one single core machine, one dual core and one quad core machine. The single and the quad core machines were virtual machines; they were configured all at the same setting, except for a different number of cores. Table 2 shows the technical specification of the three test platforms. Our aim was to test the following three issues of the content-centric service implementation:

a) Since the content-centric access layer transforms a single query from the application into several queries on the underlying storage layer, the question is how the query parallelization of "atomic queries" to the storage layer can improve the latency for processing the application query.

b) When querying for the metadata of an object, the accessing CouchDB storage was done by two operations. One operation allows querying a single value for a single key of an object. The second operation allows also querying all the key-value pairs assigned to an object at the same time. Tests should show that querying for the entire metadata set results in shorter latency than a couple individual queries.

c) And last but not least, how well does the implementation support hardware parallelism? For this we have similar setups with single, dual and quad core CPUs.

The case used represents an application from the media domain. For a video archive application, the users want to search for videos that are grouped by the producing (news) channel within a date range. The results grouped by channel are sorted by the number of accesses, which indicates their popularity. With this application, a media producer can see for a given date range, which video (news) were most popular from which channel. Note that a broadcasting organization maintains different channels. From a technical point of view, the test can be divided into the following steps:

a) The video archival application queries to the content-centric service for video material for the given data range, grouped by channel and sorted by number of accesses.

b) The service processes this query and searches for all video material for the given data range.

c) For each video, the CouchDB is queried for the channel, the title and the number of access from the video material object in the storage.

Note that the API of the underlying data store allows only for querying for a list of items in the store according to a single metadata item. Therefore, for the resulting

list of data objects, additional metadata key-values must be queried in a separate step. In particular, the underlying data store does not support "select column Y,X where ..."-type of statements.

The test shows three test cases: one querying for video material submitted for one day, one for video material submitted for two weeks and one for video material of four weeks. Since this resulted in particular queries to the content-centric service, each query string was reused for the different test setups. For the queries to the underlying storage, the system either used sequential querying or parallel querying based on the Java thread pool implementation. The metadata material consists of 45,000 data sets. Each video material item in the store has roughly about 1 kB of metadata distributed in 18 key-value pairs.

5.6. Test results

We conducted the tests on three test platforms as explained in Table 2. The measured times in milliseconds are the average mean of 20 runs. For the mean calculation, the longest and shortest run of the measured execution times were omitted. While the client submitting the query ran on a remote machine, the entire content-centric software stack with the underlying storage layer was installed onto a single test machine. The rationale for this was to test as a first step, how well hardware parallelism is supported and how well the implementation scales with larger volumes. The test results lead to the following conclusions:

a) Query latency/parallel queries. Clearly, the thread pool allowing for parallel submitted queries seems to reduce the overall execution time. Even running on just one logical core, the use of the thread pool improved executions times by 30% to 50%.

b) Single value query vs. entire object metadata at the same time. When having the choice between getting the entire meta-data set of a data item at once and querying individual metadata key-value fields individually, our results show that getting the entire 1 kB metadata at the same time is noticeably quicker that querying even three metadata fields of several bytes individually. Presumably this observation holds until the fetched metadata does not fit into one TCP/IP packet.

c) Hardware parallelism. The results taken from the single and dual core machines indicate a notable performance improvement from the use of hardware parallelism. It must be noted that the Xeon-based server hardware of the virtual machine provides better technology (caching, front side bus, etc.) than the developer machine featuring the Intel Core2Duo T6600. From the results of the virtual machine in a single-core configuration and the four cores assigned, the performance increase from using parallel threads improved by the factor 3 to 4. This indicates that the implementation scales sufficiently with a rising number of execution cores.

Table 2. Overview of test platforms

Designation, processor	Cores	Clock speed	L2 Cache	RAM	OS	Storages
T6600: Intel Core2Duo	2/2	2.20Ghz	2MB	2GB RAM	32bit Win 7	Seagate Momentus 5400.3 120GB HD
Virtual 1: QEMU Virtual cpu64	1/1	2.13Ghz	4MB	8GB RAM	64bit RHEL	500GB HD
Virtual 4: QEMU Virtual cpu64	4/4	2.13Ghz	4MB	8GB RAM	64bit RHEL	500GB HD

Source: own elaboration.

Table 3. Performance results in milliseconds Source: own elaboration

Case	Variant	Amount of datasets	T6600 Single thr.	T6600 Threadpool	Virtual 1 Single thr.	Virtual 1 Threadpool	Virtual 4 Single thr.	Virtual 4 Threadpool
One day	All Metadata	30	312	214	417	309	544	74
	Single Queries		561	377	574	392	959	175
Two weeks	All Metadata	662	6.028	3.321	7.255	3.949	9.881	1.360
	Single Queries		11.575	7.263	12.624	6.684	18.248	2.320
Four weeks	All Metadata	969	9.001	5.015	10.166	5.698	14.199	1.761
	Single Queries		17.183	10.903	19.458	9.375	26.317	3.925

Source: own elaboration.

6. Related work

There are different approaches to building a large-scale object-based storage. One way is to use an object-oriented database server, such as Versant Store [Versant] and Objectivity/DB [Objectivity DB]. Such database servers provide many concepts that are also envisaged for VISION Cloud. However, the analysis of the aforementioned products has revealed that a high-performance and scalable solution seems to be accomplished with a rather high technical effort, whereas the approach of VISION Cloud is based on horizontal "cloud-like" scaling of less sophisticated nodes.

Major cloud platform providers have their products for the storage of large data objects such as Amazon S3 [Amazon Simple…] and Microsoft Azure Blob Service [Microsoft Azure…]. These services provide some primary support for the management of large data objects or "blobs". A blob can be understood as a file having additional metadata. Some metadata is predefined, such as HTTP-metadata; however, users can also attach user-defined metadata. The use of metadata by current cloud providers is rather rudimentary. For example, there are no means to search for blobs with certain metadata, thus failing to provide a content-centric access to data. Therefore, a metadata storage requires proprietary implementation, potentially based on Azure Table storage or Amazon SimpleDB to accomplish this task.

The combination of Apache HBase [Apache HBase] and Hadoop [Apache Hadoop] offers a data store that provides many features for a distributed and replicated environment. However, the HBase interface is more oriented to a table storage and thus is not on the same technical level as the content-centric approach that we propose. Our approach offers more functionality towards the application data models, such as the relations between data objects or the interpretation of metadata, for example, as provided with the metrics service.

EMC Atmos [EMC Atmos] is an object store that supports user metadata and provides query by metadata key. However, its interface for metadata setting and query does not offer nearly as much functionality as our content-centric service. In particular, it does not provide query by metadata key and value, support for relations between objects, a metadata import service or a metrics service for queries over geo-coordinates.

Content-addressable storage (CAS) systems, such as EMC Centera [EMC Centera] and Venti [Quinlan, Dorward 2002], assign a unique name to every stored object that is produced based on the contents of the object. This makes the location of the object irrelevant, since it can be retrieved solely based on its unique name. CAS systems are tailored to archiving data that does not change often and therefore not suitable as general-purpose storage solutions.

Our approach resembles the idea of content-centric networking as proposed by [Jacobson et al. 2009]. Content-centric networking, as a part of the Future Internet movement, postulates a paradigm shift from addressing by location (i.e. URIs) to addressing by content. As such, users of the Future Internet will use content identifiers

for accessing content and the network infrastructure will determine the actual location of the requested content transparently to the user. Proposals that revolve around the content-centric idea on the network layer include DONA [Koponen et al. 2007], TRIAD [Gritter, Cheriton 2011] and ROFL [Caesar et al. 2006].

What is closest to our proposal is the research effort CIMPLE [Delaet, Joosen 2009]. Every content item is represented with a unique key that is calculated from the content itself and associated with a set of attributes that contain information regarding the owner of the content and the key used to create the unique ID. Moreover, metadata can be associated with every item that can be used to carry out search operations on the stored content. Metadata is expressed through RDF, queried through SPARQL and stored in a separate database. While CIMPLE represents a stand-alone system, the Content-Centric Service is a REST component embedded into the software stack of the VISION Cloud project. The software stack covers also other different functionalities, such as secure access, compliance modules, monitoring, billing and other areas. The Content-Centric Service makes use of cloud-like document storage to provide a high level of performance and reliability.

7. Conclusions

In this paper we have introduced an approach for content-centric access on a cloud-scale infrastructure level. The proposed approach enables access to data objects based on information about their content and relationships rather than details of underlying storage containers. Thus, it facilitates the development and identification of content networks. The information is managed by rich metadata values, which allows for scalability through the management of objects, containers, relationships and corresponding properties. Furthermore, it provides interoperability by supporting the federation of multiple disparate content networks. And last but not least, it enables management optimization for domain-specific storage operations based on the metadata values, such as popularity or importance of content, less used or unused items that can be deleted, for example.

While today most of the storage solutions add expensive hardware to address the challenges of providing knowledge to the users about what has been stored, the proposed approach enhances the user experience by enabling retrieval and other actions based on information about the stored content. Clearly, some parts of the content-centric approach already exist. For example, domain-specific solutions have been built that demonstrate some of the ingredients of the innovation, e.g. the way YouTube provides its users with access to the video content. However, these previous solutions do not exist as domain-independent software available to any application developer. Thus, our primary innovation is that we show how to abstract domain-specific ideas and use them to build a general-purpose solution for content-centric storage.

We have also demonstrated the proposed approach through a real-world application scenario, a media application. In the specific domain, the search for content is time-consuming and thus reduces the efficiency of business workflows, such as the production workflow. The engineering of the content-centric storage creates a storage API that offers application developers efficient access to content, leading to innovative functions for querying data or modeling relations between data entities.

Acknowledgment

The research leading to these results is partially supported by the European Community's Seventh Framework Program (FP7/2001-2013) under grant agreement No. 257019 – VISION Cloud Project.

References

Amazon Simple Storage Service, http://aws.amazon.com/s3/.

Apache Hadoop, http://hadoop.apache.org/.

Apache HBase, http://hbase.apache.org/.

Brewer E.A., Towards robust distributed systems (abstract), [in:] *Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing*, ser. PODC'00. New York, NY, USA: ACM, 2000, p. 7, http://doi.acm.org/10.1145/343477. 343502.

Caesar M., Condie T., Kannan J., Lakshminarayanan K., Stoica I., ROFL: routing on flat labels, *SIGCOMM Computer Communication Review* 2006, Vol. 36, pp. 363–374, http://doi.acm.org/10.1145/1151659.1159955.

Delaet T., Joosen W., Managing your content with CIMPLE – a content-centric storage interface, [in:] *IEEE 34th Conference on Local Computer Networks, 2009*,LCN 2009, pp. 491–498.

EMC Atmos, http://www.emc.com/storage/atmos/atmos.htm.

EMC Centera, http://www.emc.com/products/ family/emc-centera-family.htm.

Gritter M., Cheriton D.R., An architecture for content routing support in the Internet, [in:] *Proceedings of the 3rd Conference on USENIX Symposium on Internet Technologies and Systems*, Vol. 3, ser. USITS'01, USENIX Association, Berkley 2001, p. 4, http://dl.acm.org/citation.cfm?id=1251440.1251444.

IDC, Digital Universe Study, May 2010, http://gigaom.files.wordpress.com/2010/05/2010-digital-universe-iview_5-4-10.pdf.

Jacobson V., Smetters D.K., Thornton J.D., Plass M.F., Briggs N.H., Braynard R.L., Networking named content, [in:] *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT'09,ACM, New York2009, pp. 1–12, http://doi.acm.org/10.1145/1658939.1658941.

Jaeger M.C., Messina A., Lorenz M., Gogouvitis S.V., Kyriazis D., Kolodner E.K., Su X., Bahar E., Cloud-based content centric storage for large systems, [in:] M. Ganzha, L. Maciaszek, M. Paprzycki (Eds.), *Proceedings of the Federated Conference on Computer Science and Information Systems FedCSIS 2012*, Polskie Towarzystwo Informatyczne, IEEE Computer Society Press, Warsaw, Los Alamitos, CA 2012, pp. 987–994.

Kolodner E.K., Tal S., Kyriazis D., Naor D., Allalouf M., Bonelli L., Brand P., Eckert A., Elmroth E., Gogouvitis S.V., Harnik D., Hern´andez F.,Jaeger M.C., Lakew E.B., Lopez J.M., Lorenz M., Messina A., Shulman-Peleg A., Talyansky R., Voulodimos A., Wolfsthal Y., A cloud environment for data-intensive storage services, [in:] *CloudCom*, 2011, pp. 357–366.

Kolodner E.K., Shulman-Peleg A., Naor D., Brand P., Dao M., Eckert A., Gogouvitis S.V., Harnik D., Jaeger M.C., Kyriazis D., Lorenz M., Messina A., Shribman A., Tal S., Voulodimos A., Wolfsthal Y., Data intensive storage services on clouds: Limitations, challenges and enablers, [in:] D. Petcu, J.L. Vazquez-Poletti (Eds.), *European Research Activities in Cloud Computing*, Cambridge Scholars Publishing, 2012, pp. 68–96.

Koponen T., Chawla M., Chun B.-G., Ermolinskiy A., Kim K.H., Shenker S., Stoica I., A data-oriented (and beyond) network architecture, *SIGCOMM Computer Communication Review* 2007, Vol. 37, pp. 181–192.

Microsoft Azure Blob Service API, http://msdn.microsoft.com/en-us/library/dd135733.aspx.

Objectivity DB, http://www.objectivity.com/ pages/objectivity/default.asp.

Quinlan S., Dorward S., Venti: A new approach to archival storage, [in:] *FAST'02*, 2002, pp. 89–101.

Versant, http://www.versant.com.

VISION Cloud Project Consortium, "Project Website", http://www.visioncloud.eu/.

CONTENT-CENTRIC STORAGE: METODA UDOSTĘPNIANIA OBIEKTÓW POPRZEZ METADANE I RELACJE

Streszczenie: W artykule przedstawiono metodę organizacji zapisu obiektów w chmurze obliczeniowej mającą na celu ułatwienie przechowywania, wyszukiwania, skojarzenia, analizy i rozpowszechniania danych i treści. Obiekty zawierają często bogate opisy, ustrukturyzowane i zmienne metadane, które umożliwiają ich wyszukiwanie na podstawie ich wartości wynikającej z treści, jak również udzielanie informacji na temat znaczenia danego obiektu i jego relacji z innymi obiektami. W niniejszym artykule przedstawiono metodę zapisu obiektów wykorzystującą jego zawartość (*content-centric storage*), która umożliwia aplikacjom dostęp do obiektów poprzez informację o ich zawartości, a nie poprzez ich ścieżki dostępu w strukturze hierarchicznej. Aplikacje zatem nie potrzebują żadnej wiedzy na temat organizacji przechowywania danych lub ich miejsca w hierarchii pamięci. W artykule pokazano prototyp działania metody i oceniono jego skuteczność przy użyciu rzeczywistych scenariuszy z produkcji mediów.

Słowa kluczowe: przechowywanie w chmurze, dostęp do obiektów sterowany zawartością, metadane, modelowanie danych, przechowywanie obiektów, systemy rozproszone.