Proceedings of the Federated Conference on
Computer Science and Information Systems pp. 987–994

ISBN 978-83-60810-51-4

Cloud-Based Content Centric Storage for Large Systems

Michael C. Jaeger*, Alberto Messina†, Mirko Lorenz‡, Spyridon V. Gogouvitis§, Dimosthenis Kyriazis§, Elliot K. Kolodner¶, Xiaomeng Su‖ and Enver Bahar*

*Siemens AG, Corporate Technology
D-80200 Munich, Germany
michael.c.jaeger,enver.bahar@siemens.com

¶IBM Haifa Research Lab
Haifa, 31905, Israel
kolodner@il.ibm.com

§National Technical University of Athens,
Heroon Polytechniou, 15773 Athens, Greece
spyrosg,dkyr@telecom.ntua.gr

‡Deutsche Welle
Kurt-Schumacher-Str. 3
D-53110 Bonn
mirko.lorenz@googlemail.com

‖Telenor R&I
NO-7004 Trondheim, Norway
xiaomeng.su@telenor.com

†RAI Radiotelevisione Italiana
Centre for Research and Technological Innovation
Corso Giambone 68, I-10135 Turin, Italy
a.messina@rai.it

***Abstract*—Content centric storage refers to a paradigm where applications access data objects through information about their content, rather than their path in a hierarchical structure. The application does not need any knowledge about the data store organization, or the place in a storage hierarchy. Rather, the application is able to query for the desired content based on metadata associated with the data objects.**

We illustrate the need for content centric storage and access through an example from a media production application. Our approach for building an industrial-capable content centric storage employs a cloud storage technology where metadata is treated as first class citizen and extends this technology with an API layer that manages and leverages metadata regarding content.

I. Introduction

THERE is a growing need for digital storage. Different kinds of applications face this need. Media authoring applications work with video files, where recording format sizes are still growing beyond high definition video: Ultra HD represents the next generation video format which defines sizes up to 7680 by 4320 pixels. Server applications maintain multiple snapshots of database files. There are archives for virtual machine disks, which is an emerging market given the virtualization trends in today's enterprise IT. Furthermore, multimedia content captured and produced by mobile devices is growing exponentially. All these contribute to an ever growing number and size of data objects. 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 through any device. All these applications require powerful storage systems capable of handling large files or/and a large number of files. By handling, we mean the ability to easily ingest content of any type, to quickly find the desired content, and to smoothly access the content through desired devices. These functionalities go beyond plain data storage functionality as offered by cloud storage providers today.

Traditionally, content is put into files and organized by a hierarchy or structure. Then, the user navigates this structure when searching for particular content. When the amount of data is huge, it gets ever more difficult to maintain such a hierarchy and correctly allocate a file so that one can easily find it later. In our approach to content centric storage, the user is not limited to using hierarchies to find his 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 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.

Today using metadata in a storage system is often limited. If one wants to build a smart cloud storage solution, the ability to handle metadata from a variety of sources is therefore crucial. An integrated mobile application helps users to reveal directions to the best rated restaurants based on thousands of recommendations, pictures and videos; a media company could re-use, re-combine and re-publish existing content to

explain complex issues. Technically, such a vision must be addressed by overcoming some problems that hinder this smart re-combination and re-use of existing data objects.

The content centric storage proposed in this article is part of the large EU project VISION Cloud [1], [2], which aims at developing a cloud storage system that allows efficient and federated store of all types of data. Unlike Amazon S3, Microsoft Blob Service or hardware appliances, VISION Cloud supports also private cloud installations and does not require the use of specialized hardware.[1]

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. The paper is structured as follows: We explain the motivating scenario in Section II and derive the corresponding requirements in Section III. Section IV introduces the particular effort resulting from requirements that come from different domains and explains our solution for such a setup. This section also presents results from performance tests. Finally, the related work is presented in Section V and brief conclusions are given in Section VI.

II. Illustration of Concepts

When we look for an initial set of requirements, it is useful to first look at similar solutions from specific domains. Popular Web applications already provide examples for the content centric approach to accessing data, e.g., the aforementioned YouTube site. Another example is the Amazon online store that demonstrates some of the anticipated functionality and also 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. Figure 1 outlines the various elements that Amazon provides 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) Search for keywords 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 for 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) 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.
3) A 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) 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) A preview is shown. 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) 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;

Fig. 1. An Example for the Idea for the Content Centric Approach: Relevant Information for Picking an Item at Amazon.com.

[1]In addition to content centric storage, VISION Cloud introduces more innovations (e.g., computational storage) for cloud storage systems. More information can be found at the project's website [3].

but taken over a very large number of sessions this information could be relevant.

7) Amazon reports items that were searched 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.

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 the number of clicks to ultimately find a video that suits his/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:

- 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.
- 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.

III. Requirements

The gathering of requirements for content centric storage has been based on four distinctive use cases of the VISION Cloud project led by industrial partners: telecommunications, media, health care and enterprise. Here we illustrate the process through an example of content centric storage for media production.

The content centric storage API shall enable the user to access the data objects not only by their location, i.e., file path but also through the specification of content-related features. An application can query for data objects by information about their content, which is expressed as metadata elements. Metadata can be multi-faceted and extremely useful:

- Information about the content includes what users will see, read, or hear when opening a data object. For a document, this could be a title. For videos this could be tagging of each scene.
- Using the content centric approach provides additional opportunities as well, like understanding which video material is downloaded most. Combined ”popularity data” (e.g., metadata describing aspects such as ”likes on Facebook”, ”most commented”, and ”most viewed”) could provide powerful new filters for content.
- 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 location-based search given a 2D metric system on spatial coordinates.
- Information about relations between objects. For example, metadata can be used to indicate that an object belongs to an ordered or unordered set.

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

First, the metadata ingestion must be automated, because the lower the hurdles for adding metadata are, the richer the metadata will be that is added to the data objects. For example, considering the ingestion of objects from existing sources, Table I shows metadata fields associated with objects from YouTube, Flickr and Twitter. Simple parsing tools can extract this metadata from these sources. This provides the basis for a very powerful approach to leveraging existing metadata. A more complex algorithm could analyze which indicators (likes, views, dislikes, number of comments, rating) identifies just the really good videos, photos, and tweets, filtering out everything else. 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.

A more advanced approach for reusing existing metadata can be achieved through the preservation of expressed relationships among the objects, i.e., the objects' *structural information*. In fact, relations among objects are an important type of metadata in many professional and consumer scenarios. For example, craft editing is typically done externally, through the use of dedicated software; thus, when exporting the material to the editing workstation, it is of vital importance to preserve all the structural relationships among the various pieces (e.g., preserving the fact that tracks are part of the material, that shots and effects are parts of the video track) in order to maximize storage efficiency and, at the same time, the quality of 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:

- For example, in media production it is required to repre-

TABLE I
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 been added	Screen_name	geo_enabled
Description	Dislikes		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

sent 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.

- 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).
- 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 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 he/she uses.
- Subsumption relations are required when there is inheritance between classes of objects, and there is the 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.

IV. DESIGN AND IMPLEMENTATION

Our content centric storage system is comprised of several modules based on our analysis of the VISION Cloud use cases. Each module offers a self-contained set of functionality. Figure 2 summarizes the modules of the interface:

- The basic object services allows putting new data objects and getting existing data objects, similar to conventional object storage services, such as Amazon S3. This module wraps the interface of the underlying object service, which provides basic operations on objects and their metadata.

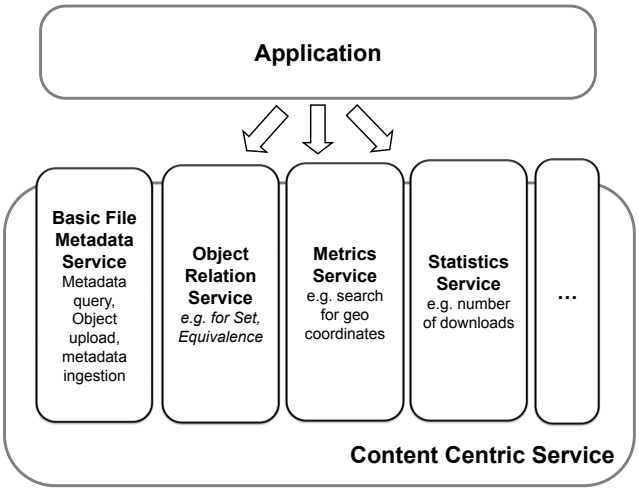

Fig. 2. Modules of the Content Centric Service.

- The relations service provides various ways to handle objects based on their metadata. This includes querying for objects by metadata keys and values. Beyond 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.
- 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 (2-dimensional) coordinates to an object. Then, a query can identify data objects that are in the vicinity of a defined geographic coordinate.
- 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.

A. Metadata Ingestion

The key element for the metadata approach is the reuse of existsing 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 XML document can be easily extracted from the used file formats. The implemented approach works as follows:

1) 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.
2) The definition of an XML Schema also defines how to process information about relations between data objects using special XML tag attributes.
3) The client uploads this schema to the data store, technically as other plain data object to the store.
4) The client assignes a schema identification to the uploaded XML schema data object in order to reference this identification at later metadata imports.
5) 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 reuse of existing metadata. Different sources are possible, for example, the import of YouTube metadata associated with a YouTube video. Other representations than XML requires the conversion to XML. Existing XML documents containing object metadata can be annotated and then uploaded in larger batches for import into the storage.

B. Architecture

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

Architectural challenges also arise for handling semantic information since VISION Cloud is designed to be a highly distributed system with high performance, scalability and availability. Thus, the CAP theorem [4] 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 an application schema or an application ontology any two out of the three properties:

1) consistency,
2) availability and
3) 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 for a metadata schema for semantic information by the application, the application using VISION Cloud must be able to deal with eventual consistency.

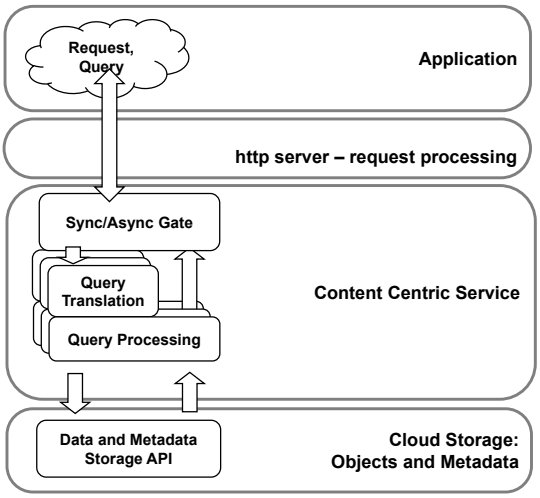

Fig. 3. Content Centric Service Implementation: Runtime View.

C. Runtime view

The basic design approach for the content centric modules is critical as it directly impacts the performance of applications working with the storage. The basic 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 [1]).

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) cloud storage.

So, the content centric modules translates queries posed via the rich query API into basic metadata queries for the key-value pairs that are supported internally by the metadata catalog. Considering a cloud-paradigm for the design of applications, the processing of this translation and the re-query operations are designed in a asynchronous and parallelizable way. Figure 3 outlines the general approach. Using a hybrid sync-async pattern 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 content centric API parses this query and creates several individual queries to the underlying storage. This storage allows for querying for one key-value pair at once, only. When all results from the individual queries are received, the intersection from the results is generated and returned to the client.

D. Implementation

Figures 2 and 3 hint at the interface for content centric storage. It offers a HTTP-based RESTful service interface to

the applications, allowing the 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 a NoSQL 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 as follows: 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 a 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.

E. 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 servlet container at version 7.0.26. The REST services were implemented using the Jersey framework. The service accessed an underlying NoSQL object storage as persistence layer. For the tests, which aimed at the content centric service the CouchDB version 1.1.0 database server has been used.

We have had three test platforms: 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 the same stetting except from a different number of cores. Table II shows the technical specification of the three test platforms. Our aim was to test the following three issues of the content centric service implementation:

- 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 query parallelization of ”atomic queries” to the storage layer can improve the latency for processing the application query.
- When querying for the metadata of an object, the accessing CouchDB storage was done by two operations: One operation allows for querying a single value for a single key of an object. The second operation allows also for querying all key-value pair assigned to an object at once. Tests should show, that querying for the entire metadata set results in shorter latency than a couple individual queries.
- And last but not least, how well does the implementation support hardware parallelism? For this have similar setups with single, dual and quad core CPUs.

The use case 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 goes into the following steps:

1) The video archival application queries to the content centric service for video material for given data range, grouped by channel and sorted by number of accesses.
2) The service processes this query and searches for all video material for the given data range.
3) 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 1kB of metadata distributed in 18 key-value pairs.

F. Test Results

We have conducted the tests on three test platforms as explained in Table II. 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 have been 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 indicate the following conclusions:

- **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%.

TABLE II
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

TABLE III
PERFORMANCE RESULTS IN MILLISECONDS.

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

- **Single value query vs. entire object metadata at once:** When having the choice between getting the entire metadata set of a data item at once and querying individual metadata key-value fields individually, our results show that getting the entire 1kB metadata at once 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.
- **Hardware parallelism:** The result 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.

V. RELATED WORK

There are different approaches for building a large scale object-based storage. One way is to use an object database such as Versant Store [5] and Objectivity/DB [6]. Object database servers provide many concepts that are also envisaged for VISION Cloud. However, the analysis of the aforementioned products has revealed that high-performance, fault-tolerant and scalable solutions seem to be accomplished with a rather high technical effort whereas the approach of VISION Cloud is based on scaling.

Major cloud platform providers have their products for the storage of large data objects such as Amazon S3 [7] and Microsoft Azure Blob Service [8]. 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 is no means to search for Blobs with certain metadata, thus failing to provide a content centric access to data.

The combination of Apache HBase [9] and Hadoop [10] 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 not on the same technical level as the content centric approach 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 [11] is an object store that supports user metadata and provides query by metadata key. However, its the 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 upload service, or a metrics service for queries over geo-coordinates.

Content-addressable storage (CAS) systems, such as EMC Centera [12] and Venti [13], 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 for archiving data that does not change often and therefore not suited as general purpose storage solutions.

Our approach resembles the idea of content centric networking as proposed by [14]. 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 [15],

TRIAD [16] and ROFL [17].

Closest to our proposal is CIMPLE [18], a research effort. Every content item is represented with a unique key that is calculated from the content itself and it is associated with a set of attributes that contains information regarding the owner of the content, the key used to create the unique ID etc. Moreover, metadata can be associated with every item that can be used to carry out search operations on the stored content. Metadata are 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 Content Centric Service makes use of a Cloud-like data storage to provide a high level of performance and reliability.

VI. Conclusions

In this work we presented a way to enhance plain object storage, primarily by associating metadata with objects and leveraging this metadata for more efficient access. Our work also provides the following advantages:

- Content-centric storage provides a chance to reduce costs over time by supporting efficient search over large stores. As a tangible consequence, it improves the availability, performance and stability of the storage system. This in turn enhances the user experience. Using content centric concepts overall costs can be reduced as retrieval, federation and other actions can be handled based on knowledge about what has been stored. Today, most storage solutions solve this problem by adding expensive hardware.
- The use of metadata and the approach of content "centricness" enable positive qualities for a storage system: it allows the system to optimize its operation based on the popularity or importance of the content (e.g., by eager fetching). Thus, content centric storage can perform more quickly and more efficiently in terms of resource usage.
- Last but not least, in the long run, this enables also more efficient maintenance of the storage in terms of identifying less used or unused objects. Capturing the popularity of content enables users to more easily identify items that can be deleted, e.g., because they are incomplete, never used or accessed and thus simply clogging up the system. Providing reliable information about what can be deleted is a novel approach supported by content-centric concepts.

In the media 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.

Clearly, some parts of the content-centric innovation 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.

Acknowledgment

The research leading to these results is partially supported by the European Community's Seventh Framework Programme (FP7/2001-2013) under grant agreement nr. 257019 - VISION Cloud Project.

References

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

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

[3] VISION Cloud Project Consortium, "Proejct Website," Accessible online at: http://www.visioncloud.eu/.

[4] E. A. Brewer, "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, pp. 7–. [Online]. Available: http://doi.acm.org/10.1145/343477.343502

[5] "Versant," http://www.versant.com. [Online]. Available: http://www.versant.com/

[6] "Objectivity DB." [Online]. Available: http://www.objectivity.com/pages/objectivity/default.asp

[7] "Amazon Simple Storage Service," http://aws.amazon.com/s3/. [Online]. Available: http://aws.amazon.com/s3/

[8] "Microsoft Azure Blob Service API." [Online]. Available: http://msdn.microsoft.com/en-us/library/dd135733.aspx

[9] "Apache HBase." [Online]. Available: http://hbase.apache.org/

[10] "Apache Hadoop." [Online]. Available: http://hadoop.apache.org/

[11] "EMC Atmos." [Online]. Available: http://www.emc.com/storage/atmos/atmos.htm

[12] "EMC Centera." [Online]. Available: http://www.emc.com/products/family/emc-centera-family.htm

[13] S. Quinlan and S. Dorward, "Venti: A New Approach to Archival Storage." in *FAST'02*, 2002, pp. 89–101.

[14] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, ser. CoNEXT '09. New York, NY, USA: ACM, 2009, pp. 1–12. [Online]. Available: http://doi.acm.org/10.1145/1658939.1658941

[15] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 37, pp. 181–192, August 2007.

[16] M. Gritter and D. R. Cheriton, "An architecture for content routing support in the internet," in *Proceedings of the 3rd conference on USENIX Symposium on Internet Technologies and Systems - Volume 3*, ser. USITS'01. Berkeley, CA, USA: USENIX Association, 2001, pp. 4–4. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251440.1251444

[17] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, and I. Stoica, "ROFL: routing on flat labels," *SIGCOMM Comput. Commun. Rev.*, vol. 36, pp. 363–374, August 2006. [Online]. Available: http://doi.acm.org/10.1145/1151659.1159955

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