A Cloud Environment for Data-intensive Storage Services

Elliot K. Kolodner[1], Sivan Tal[1], Dimosthenis Kyriazis[2], Dalit Naor[1], Miriam Allalouf[1], Lucia Bonelli[4], Per Brand[5], Albert Eckert[6], Erik Elmroth[3], Spyridon V. Gogouvitis[2], Danny Harnik[1], Francisco Hernandez[3], Michael C. Jaeger[6], Ewnetu Bayuh Lakew[3], Jose Manuel Lopez[8], Mirko Lorenz[7], Alberto Messina[9], Alexandra Shulman-Peleg[1], Roman Talyansky[10], Athanasios Voulodimos[2], Yaron Wolfsthal[1]

[1] IBM Haifa Research Lab, Haifa, Israel
[2] National Technical University of Athens, Iroon Polytechniou 9, Athens, Greece
[3] Umea University, Department of Computing Science, Sweden
[4] Engineering Ingegneria Informatica SpA - R&D Labs, Via S. Martino d Battaglia 56, Italy
[5] SICS, S-164 28 Kista, Sweden
[6] Siemens AG Corporate Technology, Otto-Hahn-Ring 6, 81739 Munich, Germany
[7] Deutsche Welle, 51063 Koln, Germany
[8] Telefonica I+D, Distrito C, Edificio Oeste 1, Ronda de la Comunicacion, Madrid, Spain
[9] RAI - Centre for Research and Technological Innovation, Corso E. Giambone 68, Torino, Italy
[10] SAP Research Israel, Hatidhar 15, Raanana, Israel

kolodner@il.ibm.com, sivant@il.ibm.com, dimos@mail.ntua.gr, dalit@il.ibm.com, miriam.allalouf@gmail.com, ewnetu@cs.umu.se, lucia.bonelli@eng.it, perbrand@sics.se, albert.eckert@siemens.com, elmroth@cs.umu.se, spyrosg@mail.ntua.gr, dannyh@il.ibm.com, hernandf@cs.umu.se, michael.c.jaeger@siemens.com, josemll@tid.es, mirko.lorenz@dw-world.de, a.messina@rai.it, shulmana@il.ibm.com, roman.talyansky@sap.com, thanosv@mail.ntua.gr, wolfstal@il.ibm.com

***Abstract*— The emergence of cloud environments has made feasible the delivery of Internet-scale services by addressing a number of challenges such as live migration, fault tolerance and quality of service. However, current approaches do not tackle key issues related to cloud storage, which are of increasing importance given the enormous amount of data being produced in today's rich digital environment (e.g. by smart phones, social networks, sensors, user generated content). In this paper we present the architecture of a scalable and flexible cloud environment addressing the challenge of providing data-intensive storage cloud services through raising the abstraction level of storage, enabling data mobility across providers, allowing computational and content-centric access to storage and deploying new data-oriented mechanisms for QoS and security guarantees. We also demonstrate the added value and effectiveness of the proposed architecture through two real-life application scenarios from the healthcare and media domains.**

Keywords-Cloud computing; Storage; Data-intensive services

I. INTRODUCTION

Cloud computing offers the potential to dramatically reduce the cost of service provisioning through the commoditization of IT assets and on-demand usage patterns. Virtualization of hardware, rapid service provisioning, scalability, elasticity, accounting granularity and cost allocation models enable Clouds to efficiently adapt resource provisioning to the dynamic demands of Internet users. Nevertheless, today's rich digital environment poses new requirements and challenges towards cloud environments: mobile devices penetrate the market, cities go digital deploying sensors and actuators, users co-develop and co-innovate (e.g. Wikipedia), social media allow for content and experiences sharing. In this context, cloud environments are facing a new challenge: the explosion of personal and organizational digital data. 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 and the digital convergence of telecommunications, media and ICT.

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

New data models for storage delivery based on data objects with rich, extensible metadata and elaborated access methods are emerging, positioning *cloud-based infrastructures for storage* as the next-generation solution to address the proliferation and the reliance on data. Nevertheless, there are a number of research challenges such infrastructures need to address in order to overcome limitations related to issues such as mobility, interoperability, storage access, security, cost, energy efficiency, etc. In this paper we present the architecture of a scalable and flexible cloud environment that enables the provision of data-intensive storage cloud services, and explore the paradigm shift in storage infrastructures driven by storage cloud technologies, introducing clear benefits to information management and middleware. To realize this vision, we have identified five main enablers that are

reflected in the proposed architecture. These include: i) management of the content through data objects and associated metadata, ii) data mobility across providers through federation mechanisms and protocols, iii) computations performed close to storage through "storlets", iv) simple and efficient access to objects based on their content and relationships, regardless of their physical location, representation and type, and v) guaranteed quality of service and security through enhanced management mechanisms (e.g. monitoring and analysis framework).

The proposed architectural approach not only targets (according to the SPI cloud stack [1]) the Infrastructure-as-a-Service model, which refers to the provision of resources (e.g. computational, storage, and networking), but also the Platform-as-a-Service model, which refers to the provision of a platform and the corresponding services (e.g. monitoring, accounting and billing, resiliency mechanisms) to enable the offering of cloud-based services. As such, our approach addresses aspects that are traditionally handled by middleware, for example, the proposed enhancements related to data-access methods and data-oriented management services (e.g. SLA management), as described in Section IV.

Furthermore, by changing the interface between middleware and storage, our approach to cloud storage offers benefits not available from traditional storage. On one hand, cloud storage no longer provides traditional guarantees such as the POSIX semantics of file systems, or the latency and throughput of high-end storage devices. Furthermore, the API to storage is changing – data objects are written all at once as large blobs of data (via put and get operations), and data objects are mostly immutable, namely write-all-at-once and read-many. On the other hand, functionality traditionally handled by middleware (e.g. by content management systems) can now be handled by the cloud infrastructure as in the proposed one. The cloud can store an object's metadata along with its data, can provide Big Table services over the objects and their metadata and in the future may even be able to accept schemas over metadata. As a result, the proposed approach no longer treats storage in an agnostic manner, and when used appropriately this can be leveraged by middleware. Furthermore, via "computational storage", the proposed architecture provides a built-in and secure environment for computational tasks that are executed close to their data and can replace some web services that are traditionally provided through web application servers.

The remainder of the paper is structured as follows: Section II introduces the enablers as concepts that allow the infrastructure to facilitate and provide data-intensive services, as well as presents a short reference to current offerings. Section III discusses the proposed infrastructure, data model and architecture, while Section IV describes two real-life application scenarios from the healthcare and media domains to demonstrate the added value and effectiveness of the proposed approach. The paper concludes with a discussion of future research and potential applications for the current study.

II. ENABLING DATA-INTENSIVE STORAGE SERVICES

In this section, we describe the main enabling concepts that should be supported in an architecture for a cloud environment that provides data-intensive storage services.

A. Raising the abstraction level of storage

Storage has been accessed traditionally via two main types of interfaces: block and file. The low level block interface enables basic read and write operations, and treats the storage as an unlimited array of raw bytes. The higher level file system interface treats storage as containers of semantically related bytes, with a directory structure relationship among the containers. Modern file systems also introduce extended attributes to files as a mechanism to associate a limited amount of additional metadata with a file. These two data models have been optimized for scale and performance.

Over the last decade a new model of object storage has been introduced for access to storage devices [2, 3, 22, 23]. This model has been successfully adopted for the cloud, replacing the traditional file system and adapting it to cloud scale (e.g. [4, 5, 9]). It flattens the tree hierarchy, which is no longer relevant, and relaxes semantics and consistency. The models that have emerged so far are not very rich with respect to metadata, and their security model is weak, depending on Access Control Lists (ACLs). They are also proprietary.

We propose a more powerful data model that fits the scale of the cloud (in the spirit of the newly emerging Storage Cloud standard of CDMI [6]), yet has rich metadata and access methods, and supports a strong yet flexible security model. This data model is optimized for immutable data, maintains a global namespace for the objects, supports object versioning, associates system and user metadata (in the form of (key,value) pairs) with objects and leverages a NoSQL- type table service to provide access to objects through this metadata. For example, in addition to the basic "put" and "get" operations, it supports List-By(key), and List-By(key, value -range) over collections of objects. Future extensions could include the association of a *schema* with the user metadata. The metadata allows highlighting the content so that it can better fit with the application (the storage cloud is no longer agnostic to the data, as in the case for file systems and object stores). Furthermore, the proposed data model offloads some capabilities that were traditionally provided by the middleware layer, to the storage layer. The motivation for this is twofold. From the point of view of the middleware, there is no longer a need to federate over the namespaces of multiple storage repositories or to maintain the association between metadata and data in databases external to the storage. From the point of view of the storage, it enables optimizations not previously attainable, such as collocating data and metadata and minimizing data loss by keeping the metadata and data under the responsibility of one system. Finally, the data model supports a distributed security model capable of delegation and federated identity, which is key for the cloud.

B. Data mobility and federation without boundaries

Virtualization platforms, and cloud infrastructures in particular, allow providers and users to rapidly redeploy and move resources. While this is beginning to be achievable for compute resources today (such as VMs), it is not the case for storage and data. In the absence of true data mobility, users cannot easily migrate their data across providers and thus suffer from data lock-in [8], which is one of the most significant obstacles hindering wider adoption of cloud services. Data mobility is also fundamental to addressing IT evolution and heterogeneity, as data needs to migrate between different platforms. Furthermore, this capability is also key to federation and interoperability between providers and systems.

In our architecture we propose built-in components to address the fundamental technical barriers to data mobility and federation, allowing new technologies to overcome these barriers. The architecture includes two types of building blocks: i) a layer that enables unified access to data across storage clouds, and federates sets of data objects maintained by users across administrative domains including mechanisms for federated security across clouds, and ii) built-in network optimizations to move data more efficiently and execute data transfers intelligently and securely. One such approach is the use of network deduplication, which minimizes duplicated data transfer over the network. However, as analyzed in [28, 29], a key challenge in this approach is to exploit these data reduction techniques while preserving privacy and providing proofs-of-ownership (PoWs) for the data.

C. Computational Storage

Compute and storage are usually treated as two different resources in a decoupled manner. Given that bandwidth is neither infinite nor networking costs negligible, for many applications it is better to move the computation to the data, rather than bring the data to the computation. As the cloud model has emerged, this idea of bringing compute to storage has been applied for restricted programming paradigms, e.g., MapReduce [9], or specifically for key-value storage services, e.g., Comet [26]. Also, approaches such as [25] study how to utilize resources in a large cluster by executing data-parallel programs. We propose a more general paradigm that works for every object-based storage repository.

The primary role of computation in emerging data-intensive storage environments is to serve data by analyzing, refining and transforming it, discovering correlations and relations, and reflecting this knowledge back into the data through metadata annotations and derived data structures. To enable "computational storage", the proposed architecture follows these main principles: i) computations are executed close to their data, since the size of an encapsulated computation is typically small compared to the size of the data it accesses, so generally the computation should be moved close to data (ideally co-located so no networking resources are consumed), ii) high utilization of cloud resources enabling parallelism where appropriate, iii) high-level control of computations by the users through policies without the need to start, stop or manage individual computations. For example, computations can be injected into the cloud and instructed to analyze all data objects of a given type or within a given context (as characterized through the data model).

These principles are reflected in our architecture through a programming model for these computational agents, called *storlets*, which were originally introduced in the context of digital preservation repositories [24]. *Storlets* are released into the cloud and activated by events on data; they define not only the computation, but also triggering conditions whereby storlets are activated (e.g. on the access of a given object, the creation of a data object with given metadata, and the addition of new metadata to an existing object), constraints that apply during the storlet lifetime (e.g. maximum CPU usage), input and output data objects as well as the necessary credentials to access them, and the management interface to deal with aspects such as billing and accounting.

A storlet can be very long-lived and be repeatedly activated based on its triggering conditions, performing some computation, and then becoming passive again. A runtime environment schedules and executes a storlet in a sandbox, enforcing constraints and mediating between the storlet and other platform services, e.g., for accessing, creating and modifying the metadata of data objects. The programming model subsumes traditional batch-job computations through the special case of a storlet for which the triggering condition is already met on storlet insertion. Regarding the model for fault-tolerance, when a storlet is passive it is treated like a data object (i.e., replicated) and when a storlet is active it is monitored and upon failure re-executed from its checkpointed state when last passive.

D. Content-centric Access

Content-centric storage is a new paradigm that enables access to a data object through information about its content, rather than a path in a hierarchical structure. An application does not require any knowledge about the physical location, the data store organization, or the place of an object in a storage hierarchy, rather it accesses the desired content based on the metadata associated with the object. This paradigm is similar to content-centric networking [15] and its data-counterpart CIMPLE [27], but targeted to cloud-scale systems.

Our approach, which builds on the rich data model described above in Section II.A, enables an application to query for content through various forms of metadata: i) user metadata, e.g., describing a data object's content and ii) system metadata, e.g., regarding usage (number of accesses) and query history to identify popular or well matched data objects. There is also synergy with computational storage; user metadata can be extracted by an application-specific storlet that asynchronously analyzes a data object and a later access can be based on the extracted metadata. Content-centric access also allows querying content based on key-value metadata pairs, and other kinds of information including object relations (e.g. equivalence and subsumption). It supports any domain by allowing the definition of domain-specific storage optimizations.

Furthermore, it scales to the cloud, allowing storage to be spread out across multiple clusters and data centers.

E. Capabilities for Cloud-based Storage

Although cloud storage as available from providers such as Amazon and Google offers useful features such as demand-based access to raw storage resources, it is not ready to store the critical data of individuals, businesses and governments with the required reliability and QoS as evidenced, for example by the rudimentary SLA of Amazon S3 [5, 12]. We are developing technologies necessary to close these gaps, addressing aspects of QoS and security assurance as required for business critical and sensitive applications while building a cloud infrastructure that continues to maintain the cloud spirit (e.g. virtualization, pay-per-use, scalability). Current cloud storage providers offer SLAs that guarantee service availability and give service credit or refunds for lack thereof [12, 17, 18], but do not address data availability and protection. In research an architecture and protocol for an SLA-based trust model for cloud computing [19], and approaches for managing the mappings of low-level resource metrics to high-level SLAs [20, 21] have been proposed.

Architectural challenges include: supporting multi-tenancy, where a massive number of users share the same storage infrastructure, guaranteeing secure and authorized access to the data and services, and providing tools for checking compliance with standards and regulations. In addition, given the scale, management of the storage cloud needs to be as automatic as possible, e.g., requiring the automatic placement of data, and automated provisioning and operation of the underlying storage. Furthermore, the architecture must provide hooks for accounting and billing of the storage services to be used for measuring, charging and reasoning about their cost. It also requires an advanced monitoring mechanism, going beyond a simple messaging system to aggregating, applying rules and extracting valuable information from the analysis, and to being modular, i.e., allowing new sources of information to be included as the need arises. Monitoring in cooperation with an advanced SLA management framework (which among others, take into consideration content-related terms) enables proactive SLA violation prevention.

F. Current offerings

The most notable commercial cloud storage services include Amazon S3 [5], Windows Azure Blob Service [13], EMC Atmos [7] and Google Storage for Developers [14], which do not fully realize the enablers discussed in Section II. Starting from the data models, they are basic. Amazon S3, Google Storage, and the Windows Azure Blob Service allow associating user metadata in the form of key value pairs with objects and blobs, but they simply store the metadata and pass it back. EMC Atmos has a slightly richer model; it allows some of keys (called tags by Atmos) to be listable; this enables retrieving the objects that have a specific tag. The support for federation does not exist or is limited and requires homogeneity. Amazon S3, Google Storage and the Windows Azure Blob Service do not have any support for federation. EMC Atmos allows federating data in an Atmos system in a customer data center with the customer's data in a cloud, provided it is also implemented with Atmos. No current cloud storage offering provides computational abilities as an integral part of the cloud storage system to the best of our knowledge. Access to an object is solely through its name with Amazon S3, Google Storage and the Windows Azure Blob Service. As mentioned above, EMC Atmos has a slight richer access capability through its listable tags. But no current cloud storage system has a rich flexible access to storage based on its content and relationships. Finally, the QoS mechanisms and SLAs provided by current offerings are very basic. In our approach, models, requirements and SLA schemas are expressed not only on storage resources and services, but also on the content descriptions for the underlying storage objects, in support of content centric storage.

III. STORAGE CLOUD ENVIRONMENT

In this section we present the underlying infrastructure, the data model and the architecture of the proposed storage cloud environment that addresses the challenges and realizes the concepts presented in the earlier sections.

A. Infrastructure

The storage cloud is built on an infrastructure that consists of multiple data centers, each of which may have one or more storage clusters containing physical compute, storage and networking resources. The data centers are connected by a dedicated network. The minimum bandwidth for inter-data center links is 1GB.

A *storage cluster* is composed of storage rich nodes constructed from commodity hardware and connected by commodity interconnect. As common for cloud infrastructures, the storage cloud is built from low cost components, ensuring reliability in the software, and building advanced functionality on top of this foundation. For example, given today's hardware, the initial hardware configuration for the nodes could be 4 or 8 way multiprocessors (taking multicore into account) with 12 to 16 GB of RAM. Each node could have 12 to 24 high capacity direct attached disks (e.g. 2TB SATA drives). The cluster interconnect is 1GB at a minimum. The architecture, design and implementation should support a system with hundreds of storage clusters, where each storage cluster can have several hundred nodes and the storage clusters are spread out over dozens of data centers.

B. Data Model

The data model extends the emerging cloud object models (e.g. S3 [5], Atmos [7] and CDMI [6]). At the heart of the proposed data model is the data object. A data object contains data of arbitrary type and size, and has a unique identifier that can be used to access it. Conceptually an object is fixed content - it is written as a whole and cannot be partially updated in byte ranges, but it can be partially read. An object may be overwritten, in which case the whole content of the object is replaced. Versioning is supported and when it is enabled, the system retains the previous version of

the object. Data objects are contained in containers. There is no nesting or hierarchy of containers. Each data object resides within the context of a single container. Containers serve several purposes:

- Data management. Containers group related data objects. Policies are set on a container and applied to all objects in it, e.g., whether the objects in the container are versioned.
- Isolation. Containers divide the namespace at the highest level, and provide isolation among objects between containers.
- Internal management. Containers are the unit of placement. This reduces the frequency of global placement decisions, reduces the size of location information that has to be retained globally, and helps in routing client requests efficiently to the right cluster in the cloud.

Two categories of metadata are associated with a data object: user metadata and system metadata. The user metadata is set by the user and contains information about the object. Its meaning and context are transparent to cloud storage system. However, the system does recognize the format of the user metadata and enables queries based on its content. A common format for user metadata is a list of name-value string pairs. A client can also use other formats and indicate the format through a XML schema. In contrast to user metadata, system metadata has concrete meaning to the cloud storage system. It either directs the system how to deal with the object (e.g. access control, reliability, performance requirements), or it provides system information about the object (e.g. size, creation time, last access time) to the user.

Updating the metadata for an object is possible without updating the actual data; in particular, a new metadata field can be added or an existing metadata field can be updated without updating the other fields. On the other hand, the user cannot update the data of an object without updating its metadata, i.e. when the data of an object is overwritten all of its metadata must also be replaced.

Regarding metadata and data retrieval, a user can retrieve the whole object, both its metadata and data, at the same time. In this case the system guarantees strong consistency between the object version and the metadata, i.e. the metadata returned is the metadata that belongs with that version. A user can also retrieve the metadata for an object without retrieving its data.

Containers have metadata associated with them. This metadata can be user metadata or system metadata as described earlier for objects. Naturally, containers have a different set of system metadata items, for example, a quota.

Besides the metadata for objects and containers, we extend the data model to include computation on the data objects, which is performed (executed) within the cloud storage environment. As mentioned in Section III.C, computations are performed through storlets that are triggered according to specific events.

Objects may be replicated across multiple clusters and data centers. The degree of replication and placement restriction policies are defined and associated with the object's container. We employ a symmetric replication mechanism, where any operation on an object can be handled at any of its replicas. The consistency model for updates to both objects and their metadata is eventual consistency [30]. A storlet, when triggered, is executed once, usually at the site where the triggering condition first occurred.

The account model includes tenants and users. A tenant is an organization that subscribes to storage cloud services. A tenant may represent a commercial firm, a governmental organization, or any other organization or group of persons. A user is the entity that actually uses the storage services. The term "user" may refer to a person or to an application. A user belongs to exactly one tenant, although a person could own a user account in more than one tenant. A user has a unique identifier within its tenant and has credentials allowing it to authenticate itself. A user may create containers and data objects in them. Ownership of a container is assigned to a user within the tenant. Ownership of a data object is assigned to a user within the tenant to which the object's container belongs (typically but not necessarily the container's owner).

C. Architecture

We conceptualize the architecture in two dimensions. In the first dimension, the architecture has a logical separation between data-related operations (e.g. adding, changing and deleting data) and management operations (e.g. service provisioning and monitoring). In the second dimension, each of these has an external access / interface layer that provides access to users and applications, and an internal operating layer that executes these external requests as well as autonomous operations such as dynamic optimizations, load balancing, and monitoring. Schematically, these dimensions provide a layered foundation for the architecture.

Figure 1 presents a high-level view of the architecture including the interfaces towards applications, users and administrators, the layers of the architecture and the interactions between the layers. The architecture introduces two complementary services, the Data Service and the Management Service, which together provide the functions of a cloud for data-intensive services. The Data Service, including the Data Access Layer (DAL) and the Data Operating Layer (DOL), enables manipulation of data objects and their metadata, computation on storage, mobility, availability, reliability and security. The DAL provides unified interface to data across the clusters of a cloud, and encapsulates the DOL which realizes the data service over a set of distributed heterogeneous physical resources.

The Management Service, including the Management Interface Layer (MIL) and the Management Operating Layer (MOL), enables service provisioning and monitoring, accounting and billing, security management and transformation of user-specified service level requirements to management operations on the underlying infrastructure level. This service is also distributed across the clusters of a cloud. The MIL, just like the DAL, provides a unified interface to management of services across the clusters of a cloud, and encapsulates the MOL which realizes the

management service over a set of distributed heterogeneous physical resources. The MIL deploys management models that translate business level objectives into operating level settings and tasks.

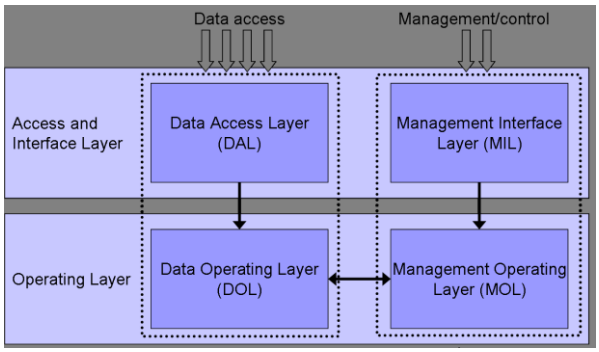

Figure 1. Layers, roles and interfaces in the architecture

The realization of the layers is distributed. For example, an object may be stored in one cluster and accessed through a request addressed to a second cluster. The DAL in the second cluster finds the target cluster on which the object resides, and transparently redirects the request to it. Furthermore, the implementation of the layers is highly distributed and parallel; the same software stack runs on every server of a cluster, and many client operations carried out in parallel on each server.

The architectural separation between the data and management services is inspired by the unique service model of the storage cloud. In compute clouds, the management service is used to provision and manage compute resources, which interact with external entities as defined by the service provider. The storage cloud is different - once the storage resources are provisioned, they may be used by different, independent service consumers through cloud storage APIs, with different characteristics and requirements on latency, throughput, availability, reliability, consistency, etc. The data service and the management service are designed to be separate and independent in order to facilitate this differentiation and provide the flexibility required to enable the innovations mentioned earlier.

In the remainder of the section we provide more details on the layers. First, though, we describe the ***Global View***, which provides common services accessed by all four layers. It is a cloud-wide service that runs on representative nodes in each of the clusters composing the storage cloud and must be highly available, despite possible node, cluster and data center failures, and partitions between clusters/data centers. It consists of three services: the Global Catalog, the Resource Map and the User Service. The Global Catalog maps from container name to the clusters where the container's replicas reside and also holds container metadata. The Resource Map holds an inventory of cloud resources: the location of clusters, the distances and bandwidth between clusters, and the resources available in each cluster. The User Service holds information about tenants, users and authentication.

1) Data Access Layer

The primary interaction point for applications and clients of the storage cloud is the Data Access Layer, which provides access to the content across the clusters of the underlying infrastructure. It includes: i) security components for access control (authentication and authorization), and ii) content-centric access components to translate complex storage requests from the application into basic queries on the underlying store in the DOL. The components are passive and act upon request. A pipelined design enables parallel processing across the underlying resources, so that the translation of the content-centric requests and the handling of the resulting queries to the DOL can be decoupled, thereby enabling the optimal assignment of processing resources.

Figure 2 shows the technical organization of the DAL. It depicts the components and the interactions between them.

The ***Request Handler*** processes incoming requests and implements basic performance and scalability features with regard to the distribution of the system. It interacts with Secure Access Control to authenticate and authorize a request, and passes the request to the proper component of the DAL for handling.

The ***Content-Centric Service*** translates complex queries posed via its rich query API into basic metadata queries on the key-value pairs associated with the objects. Employing a cloud paradigm for application design, the translation processing and the derived query operations are designed in an asynchronous and parallelizable way. Using an inverted half-sync-half-async pattern approach, queries are received from the application in a synchronous manner, while internally queries are issued asynchronously to the internal storage subsystem. This approach allows rearranging internal queries according to complexity, priority or other criteria. For this purpose the DAL and DOL are connected via an asynchronous communication component.

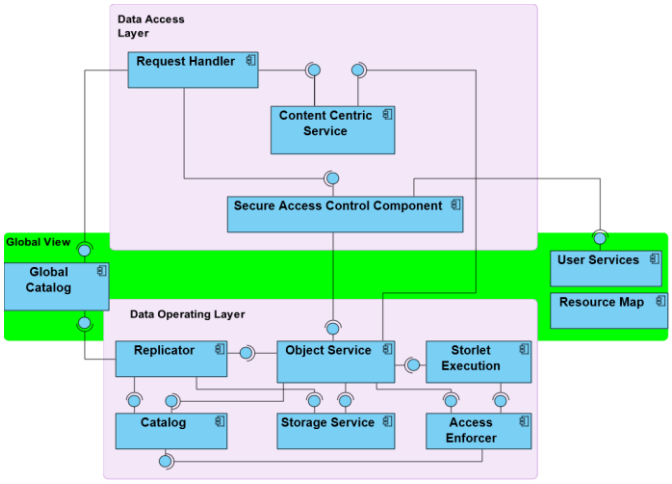

Figure 2. Data Access and Data Operating Layers

Secure Access Control provides authentication and authorization for requests. It interacts with the User Service of the Global View in order to obtain data related to users and access control.

2) Data Operating Layer

The objective of the Data Operating Layer is to provide the core object services. It acts as a "middle" layer between the DAL functionality and the storage itself. It supports both the data model and the other features of the proposed architecture such as an execution environment for storlets, data mobility and queries on metadata for content centric storage. The DOL is highly scalable and continuously available, and also provides hooks for management (e.g. for controlling QoS).

The technical organization of the DOL is shown in Figure 2. The components of the DOL are cluster-wide services that are highly distributed and have representatives that run on each node of a storage cluster. We describe them below.

The ***Object Service*** orchestrates the execution of DOL operations by interacting with the DOL's other components, e.g., for a put operation on an object, it checks authorization through the ***Access Enforcer,*** stores the object's data in the ***Storage Manager***, stores the object's metadata in the ***Catalog***, replicates the object through the ***Replication Manager***, and notifies ***Storlet Execution*** that the put has occurred.

The ***Catalog*** is a cluster-wide service that stores the metadata for each object residing in its cluster. It provides an efficient mapping from an object's name to its storage location in the cluster and its metadata. It also provides query operations over the objects in a container, such as query by key name (return all objects that have a particular key in their metadata) and query by key name and value range (return all objects that have a value within a particular range for the key). These queries are essential to the implementation of content-centric storage. The Catalog also stores state that needs to be shared across a cluster, for example the internal state of the ***Replication Manager***. The Catalog is implemented using Cassandra, a distributed column-oriented NoSQL database.

The role of the ***Replication Manager*** (RM) is to replicate data across the clusters of the system to the required degree of resiliency both during regular operation and failure recovery. During regular operation, the RM is called by the Object Service when containers or objects are created, deleted or updated. It is responsible for ensuring that the new state which results from such an operation is replicated to all other clusters where the corresponding container/object resides. In failure cases, the RM is invoked in order to restore containers or objects to the desired level of resiliency. The RM replicates both data and metadata across clusters. It is not responsible for the replication of object metadata within a cluster (this is handled by the Catalog),

The ***Storage Manager*** encapsulates the underlying storage system, providing storage for object data and also for the Catalog. It could employ a distributed file system or it could use the local file system on each server in the cluster.

Storlet Execution receives event notifications from the Object Service and triggers the appropriate storlet. Storlet Execution provides an execution environment for storlets: it stores references to passive storlets (storlets that are not currently active and stored as data objects); it activates storlets that have been triggered; and it runs active storlets in a sandbox that safely controls access to their resources, while it also monitors them. When a storlet is first stored into the DOL as a data object, Storlet Execution is also responsible for preparing it for execution, e.g., putting it in a passive state and setting up its triggers.

The ***Access Enforcer*** *i*s the component that enforces the access control decisions at the DOL layer. For example, during the storlet execution it is contacted to validate the permissions of the storlet to perform the desired operations.

3) Management Interface Layer

The Management Interface Layer provides APIs and interfaces for management, e.g., container management, account management, SLA management, and accounting and billing.. Through the MIL, users can also develop models including metadata associated with the storage resources, the service characteristics, the usage, the service requirements, etc. These models govern the operation of the MOL. Authentication and authorization mechanisms allow the various users of the platform to access the management mechanisms, while adhering to the compliance requirements. Figure 3 shows the main components of the MIL.

Tenant and User Management allows the creation and deletion of tenants and users as well as defining roles for users (e.g. customer or administrator). This information is captured in the User Service of the Global View. Through this component, a cloud administrator is able to create tenants, a tenant administrator is also able to create users that belong to it and have access to the cloud services to which the tenant has subscribed. Moreover, a tenant administrator is able to define the different roles a user may have for accessing different services.

SLA Management provides an interface to the SLA management framework residing in the MOL. Users can request an SLA template regarding specific services the platform offers, create an SLA request by setting specific quality parameters in a template, sign an SLA and receive notifications with regard to SLA violations.

Container Management allows users to create and delete containers, and update container metadata. It also allows a user to list the containers belonging to a specific tenant or user.

Model Development provides an interface to construct management models. A model incorporates various kinds of information, such as object metadata, resource and service related information, usage related data, requirements, and SLA-related information. Models express the importance of data (an object or set of objects) and a resource in comparison to others, and efficiently provide input to decision making and placement mechanisms, as well as contribute to better resource allocation and power management by the MOL.

Accounting and Billing collects raw resource usage metrics (e.g. capacity, bandwidth, CPU utilization, number of storage operations, number of objects, replication level, number of executed storlets) and SLA violation events. For each tenant it tallies resource usage and compensates for SLA violations in order to calculate a monetary amount, according to a tenant-specific billing and pricing scheme.

This component also provides user-level accounting to a tenant, so that a tenant can bill its own users.

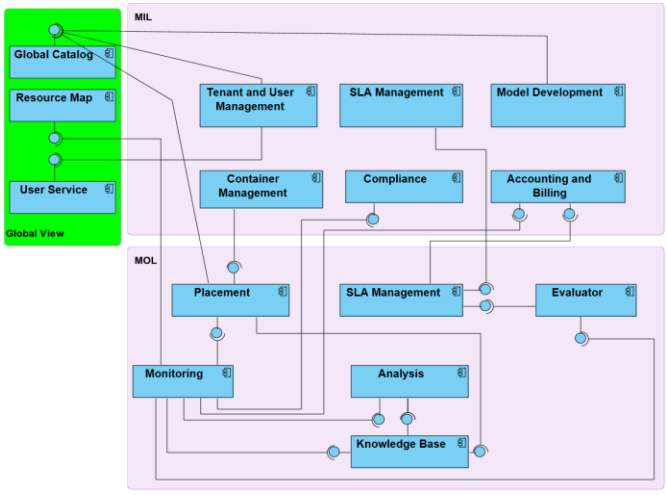

Figure 3. Management Interface and Management Operating Layers

Compliance checks operation logs and flags compliance violations, notifying the system administrator that they have occurred. It takes into consideration compliance to EU directives such as 2002/58 (personal data circulation), 2006/24 (retention of data and public communications networks).

4) Management Operating Layer

The Management Operating Layer provides management services for the proposed cloud environment. It consists of two parts: a *cloud-wide view* that has access to usage patterns across all data centers and clusters and makes decisions at the cloud level, and a *cluster view* that has a view of usage patterns within its cluster and makes decisions at the cluster level. Thus, the decision making process of the MOL's components is hierarchical: first a decision is made at the cloud level, e.g., choosing the storage clusters where data will be placed; and then a corresponding decision is made at the cluster level, e.g., to decide on which server to place the data. Figure 3 shows the components of the MOL.

A ***Knowledge Base*** stores statistics and history information with regard to previous management decisions. There is a knowledge base in the cluster view for each cluster and in the cloud view for the cloud.

Monitoring collects data from the DOL as well as from the infrastructure through low-level probes, and aggregates and distributes monitoring information across the clusters and across the cloud.

Analysis analyzes the monitoring data in order to find correlations among the various metrics, and to discover usage patterns per container/user/tenant, such as periodic bursts or performance degradations. The usage models discovered are stored in the appropriate Knowledge Base. Analysis consumes the data provided by the monitoring component and stored in the Knowledge Bases (where monitoring data is aggregated) and delivers the outcome of its analysis to the Evaluator and to the appropriate Knowledge Base (cluster or cloud level).

SLA Management consumes the analyzed monitoring data in order to enable the proactive prevention of SLA violations. It takes into account specific SLA terms as well as policies that may be set by an administrator (e.g. create an additional replica in case of a given increase in user requests) and through the Evaluator pro-actively triggers events to prevent SLA violations. A novel aspect of SLA Management are content-related terms in order to support policies regarding content-centric access.

The ***Evaluator*** receives information from two sources: i) management components (e.g. SLA management) regarding configuration parameters (e.g. a monitoring threshold), and the ii) analysis component. Based on this information, the Evaluator generates events and propagates them to the appropriate management component to perform actions accordingly.

Placement places both data and storlets. For data it decides on which clusters to place the replicas of a container, and for storlets it finds the appropriate resources for their execution. It takes into account constraints such as locality (e.g. proximity of a container replica to its users), compliance (e.g. data cannot leave the borders of a certain country), and resiliency (e.g. the number of copies and their placement in order to achieve a certain level of reliability). Regarding storlets, there is an intrinsic relationship between the placement of an object and the storlets that run on it. For example, there is a performance trade-off between choosing powerful or less loaded compute nodes for faster storlet execution and minimizing the cost of data transfer by placing the storlet closer to its data.

IV. Validation Cases

We present two real-life application scenarios from the healthcare and media domains to demonstrate the added value and the effectiveness of the storage cloud architecture.

A. Healthcare

This scenario focuses on personalized healthcare, which is a patient-centric view of all healthcare activities, related data and services. A patient's healthcare data is stored and used either by the patient or by one of his/her healthcare providers. Third party applications may also access a patient's data to provide services to the actors in the scenario. Moreover, healthcare data is commonly stored according to complex standards such as HL7, DICOM or OpenEHR. These standards provide exact descriptions of medical information while at the same time allowing healthcare systems and medical devices to interoperate reliably and precisely. Based on the above, the personalized healthcare scenario poses *requirements* on storage cloud environments that are met through the enablers and architecture described in this paper:

1. The ability to store all patient-related data as native and immutable objects and make these objects and the relationships between them accessible as a data service. Here content-centric access supports relationships and access to the objects based on their relationships. Storlets provide the means to transform data to required formats, and extract and associate appropriate metadata. Current storage offerings do

not provide such content-centric or storlet functionalities and therefore application specific services, responsible for maintaining these relationships, need to be deployed.

2. The ability to hide the complexity of healthcare data to developers and support them to supply the data as a service, while securing the data and providing an utmost level of privacy for the patient. Here the metadata and content-centric capabilities ease the task of organizing and retrieving medical data. Furthermore, storlets enable healthcare applications to efficiently perform complex transformations, aggregations and analysis on masses of healthcare data without downloading the data to an application's local storage or needing to deploy VMs that contain the necessary applications, as would be the traditional way in present Clouds. In addition to reducing bandwidth requirements, our approach has the additional benefit that it makes it easier to protect data against unauthorized use.

3. The ability to provide data to users according to their roles and context (e.g. doctors, patients, online medical service providers, researchers, pharmaceutical companies). Here, again storlets can adapt the view of the data based on the user's role, e.g., a storlet anonymizes data before providing access to researchers.

B. *Media*

This scenario focuses on professional media management in media production. Established media production workflows need to evolve to be more flexible and interoperable with standard IT. A starting point is the output of a typical media generation device (e.g. a camcorder) or tape-based media storage in the form of an MXF file. MXF is the SMPTE standard specifying the file format for exchange of audiovisual material between professional devices. An MXF file can contain several Material objects, each of which is an aggregation of audio and video Tracks. Each Track can be further decomposed into Sequences, each of which has a pointer to the actual "Essence" (i.e., the audio and/or video stream whose rendering generates the output material). Such a data structure, nowadays, is actually hidden inside the file and only dedicated software stacks are able to use it in practice. Based on the above the media production scenario poses *requirements* on storage cloud environments that are met through the enablers and architecture described in this paper:

1. The ability to avoid vendor lock-in and overcome the lack of interoperability between different devices. Here, the data model allows MXF material to be uploaded and represented as Material, Track, Essence objects, where metadata records the relationships between them, and content-centric access allows flexible ways to access the uploaded data. This enables a standard way of storing and accessing media content.

2. The ability to access, manipulate and manage material structure at the storage level. Here content-centric access allows easy browsing of the content based on its natural structure. Storlets can provide processing capabilities such as feature extraction and transcoding without the need to download the content. They can also provide the ability to extract shot and keyframe information. Overall, the ability to provide storlets allows higher-level application stacks to be thinner and easier to manage and interchange, similar to the case of healthcare discussed earlier.

3. The ability to allow users to interact with the content in order to "characterize" it with the use of tags and metadata. Here, again, the rich data model and content-centric access are a very good fit for media. While this functionality is already present in many web applications, it is application specific. In VISION Cloud it is supported in a generic fashion in storage and can be tailored for any application.

V. CONCLUSIONS

A representative figure highlighting the huge amount of data currently being produced is the fraction of data on the Internet that is indexed by Google, which is only 0.004% [16]. In this context one of the main challenges in cloud environments is the management of the data being produced by various sources (e.g. sensors, cameras), and also by people (i.e., user generated content). Management refers not only to storing but also to accessing these data, moving data across providers, performing computational tasks on data, and addressing issues related to security, quality of service, costs. In this paper we presented the architecture for a scalable and flexible cloud environment that is being developed by the VISION Cloud EU-funded research project. The architecture addresses the challenge of providing data-intensive storage cloud services through raising the abstraction level of storage, enabling data mobility across providers, allowing computational and content-centric access to storage and deploying new data-oriented mechanisms for QoS and security guarantees. We have also demonstrated the added value and effectiveness of the proposed architecture through two real-life application scenarios from the healthcare and media domains.

ACKNOWLEDGMENT

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

REFERENCES

[1] Peter Mell and Tim Grance. The NIST definition of cloud computing. Technical report, July 2009.

[2] "Object-based Storage Device Commands" (OSD), SNIA, T10 committee http://www.t10.org/ftp/t10/drafts/osd/osd-r10.pdf

[3] The eXtensible Access Method (XAM) specification, SNIA XAM Initiative http://www.snia.org/forums/xam/

[4] EMC Atmos. http://www.emc.com/storage/atmos/atmos.htm

[5] Amazon simple storage service (Amazon S3), 2009. aws.amazon.com/s3.

[6] Cloud Data Management Interface (CDMI), SNIA Cloud Storage Technical Work Group http://www.snia.org/cloud

[7] EMC Atmos storage service http://www.emccis.com/

[8] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. Above the clouds: A berkeley view of cloud computing. Technical report, University of California at Berkeley, February 2009.

[9] OpenStack Object Storage, http://openstack.org/projects/storage/

[10] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. OSDI '04. pages 137–150.

[11] "Amazon Web Services Goes Down, Takes Many Startup Sites with It", Tech Crunch, February 2009

[12] Amazon S3 Service Level Agreement, http://aws.amazon.com/s3-sla/

[13] Microsoft Windows Azure Platform http://www.microsoft.com/azure/default.mspx

[14] Google Storage for Developers. http://code.google.com/apis/storage/

[15] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca L. Braynard. Networking named content. In Proceedings of the 5th international conference on Emerging networking experiments and technologies, CoNEXT '09, pages 1–12, New York, NY, USA, December 2009. ACM.

[16] M. Jasra, "Google has indexed only 0.004% of all data on the Internet", http://www.webanalyticsworld.net/2010/11/google-indexes-only-0004-of-all-data-on.html.

[17] Nirvanix service level agreement. http://www.nirvanix.com/sla.aspx

[18] Microsoft Corporation. Windows azure pricing and service agreement, 2009. URL http://www.microsoft.com/windowsazure/pricing/.

[19] Mohammed Alhamad, Tharam Dillon, and Elizabeth Chang. Sla-based trust model for cloud computing , Network-Based Information Systems, International Conference on, pp. 321–324, 2010.

[20] Ivona Brandic, Vincent C. Emeakaroha, Michael Maurer, Schahram Dustdar, Sandor Acs, Attila Kertesz, and Gabor Kecskemeti. Laysi: A layered approach for sla-violation propagation in selfmanageable cloud infrastructures. Computer Software and Applications Conference Workshops, 365–370, 2010.

[21] Vincent C. Emeakaroha, Ivona Brandic, Michael Maurer, and Schahram Dustdar. Low level metrics to high level slas - lom2his framework: Bridging the gap between monitored metrics and SLA parameters in Cloud environments. The 2010 High Performance Computing and Simulation Conference (HPCS 2010),

[22] CAS, Content Addressable Storage, http://en.wikipedia.org/wiki/Content-addressable_storage

[23] S. Quinlan and S. Dorward. Venti: a new approach to archival storage. Proceedings of the FAST 2002 Conference on File and Storage Technologies, 2002.

[24] Simona Rabinovici-Cohen, Michael Factor, Dalit Naor, Leeat Ramati, Petra Reshef, Shahar Ronen, Julian Satran, David L. Giaretta. Preservation DataStores: New storage paradigm for preservation environments. IBM Journal of Research and Development 52(4-5): 389-400 (2008).

[25] Michael Isard, Mihai Budiu, Yuan Yu, Andrew Birrell, and Dennis Fetterly. Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks. European Conference on Computer Systems (EuroSys), Lisbon, Portugal, March 21-23, 2007.

[26] Roxana Geambasu, Amit Levy, Tadayoshi Kohno, Arvind Krishnamurthy and Henry M. Levy. Comet: An Active Distributed Key-Value Store. Proc. of the Conference on Operating Systems Design and Implementation (OSDI), October 2010.

[27] Thomas Delaet, and Wouter Joosen, Managing your content with CIMPLE - a content-centric storage interface. IEEE Conference on Local Computer Networks - LCN, pp. 491-498, 2009,

[28] Danny Harnik, Benny Pinkas, Alexandra Shulman-Peleg. Side Channels in Cloud Services, the Case of Deduplication in Cloud Storage. IEEE Security & Privacy 8(6): 40-47 (2010).

[29] Shai Halevi, Danny Harnik, Benny Pinkas, Alexandra Shulman-Peleg. Proofs of Ownership in Remote Storage Systems. CCS 2011: 18th ACM Conference on Computer and Communications Security. Oct, 2011.

[30] Werner Vogels. Eventually Consistent. ACM Queue 6, 6 (October 2008), 14-19.