Storage and Data Management in a post-Filesystem World

Preservation and Archive SIG

Austin, TX

January 2012
Born Digital data is being generated by an ever diversifying set of devices

75% of data being generated by individuals (IDC)
  - Amount of Data doubling every two years

5 Billion Mobile Phones
  - Tablets on the rise

Storage space on devices will always be limited (although continues to grow exponentially)

Device Apps are accessing content/storage via the Internet
  - Cloud Storage largely driven by device based use cases

Traditional data storage interfaces (filesystems) are not up to this task!
Distilled Requirements

- Need a Global Namespace for data location
  - FILE handle -> URL + Unique Object ID globally

- Need support for Rich Metadata associated with the data
  - Plus Metadata Query and Index based Search

- Need HTTP access – browser support

- Data Services need to be transparent to the user, but extensible and powerful
  - Driven by Metadata

- Support for large objects

- Support for active preservation operations
  - Driven by preservation Metadata
The SNIA has produced the Cloud Data Management Interface (CDMI) standard for post-Filesystem Data Storage:
- i.e. Public and Private/Hybrid Storage Clouds

Specifically designed to meet the requirements (for the foreseeable future) of contemporary data usage and management:
- Superset of features currently being offered, but “shrink to fit”

Innovative use of Metadata to express “Data Requirements”:
- Requirements are then met by the implemented Data Services

CDMI can be (and is being) extended compatibly:
- To accommodate new types of data requirements, use cases
Where is it being used?

- Commercial Cloud Storage vendors started announcing implementations in April 2011
- Increasing Adoption by eScience and Academic community
  - Especially in Europe (OpenNebula open source cloud, others)
- Quarterly Plugfests with growing participation
  - > 30 participants – Dusseldorf, Germany February 2012
- CDMI 1.0.1 on it’s way to International Standard status
  - JTC 1 PAS submission ballot underway
- CDMI has a reference implementation (Java™ under BSD)
  - Downloaded by hundreds of developers worldwide
Data Portability Standard
- Move Data (and most importantly – Metadata) from cloud to cloud

Advanced Cloud Services
- Data System Metadata allows cloud vendors to up-sell!
- Specialized storage clouds for specific use cases

Logging, Security, Audit Trails

Extensible to accommodate rapid innovation in cloud market
- Proposed Extensions: CIMI/OVF, Versioning, Jobs

Shrink-to-fit
- Only implement what makes sense for your cloud
- Profiles: Simple Storage (i.e. S3), Simple Storage Management (NAS)
Elastic demand for web based media (video, eBooks, audio)

Backup to the cloud
  - Restore, Recovery, “Seed” the backup with hard drive

Sync of files to the cloud and multiple devices
  - Internet “Drive” secondary storage

Archive/Preservation in the cloud
  - Including Compliance, Retention and eDiscovery

Storage for Cloud Computing
  - Support for legacy storage interfaces key
  - Self Service Storage Administration
Clients can be in the cloud or enterprise and provide additional services (computing, data, etc.)

Management of the Cloud Storage can be standalone or part of the overall management of your cloud computing

Clients acting in the role of Managing Data/Storage

SNIA Cloud Data Management Interface (CDMI)

SNIA Cloud Data Management Interface (CDMI)

Multiple, Proprietary Interfaces

Data/Storage Management Client

Cloud Data Management

Data Services

Storage Services

Copyright © 2010 Storage Networking Industry Association
Models for Cloud Ecology

- Cloud Federation
- Computing Cloud
  - Data Usage
  - Data Usage
  - Cloud Peering
- Object Storage Cloud
  - Cloud Peering
- Object Storage Cloud
- Distribution Cloud
  - Multiple Distribution Points
  - Cloud Peering
CDMI Basic flow:

CDMI Client issues requests

HTTPS: PUT, GET, POST, DELETE
- MimeType: application/cdmi-______
- cdmi-object, cdmi-container, cdmi-queue, cdmi-domain, cdmi-capability*
- Data, Metadata

CDMI Implementation issues response

HTTP Status (200 OK, 201 Created, etc.)
- MimeType: application/cdmi-______
- cdmi-object, cdmi-container, cdmi-queue, cdmi-domain, cdmi-capability
- Data, Metadata

*CDMI MIME Types are Registered with IANA
The resources which are accessed through the RESTful interface

- **Root**: https://<offering>
- **Capabilities**: https://<offering>/cdmi_capabilities
  - Key Value
  - Key Value
  - ...

- **Container A** - cdmi-container
  - https://<offering>/containerA
  - Key Value
  - Key Value
  - ...

- **Container B** - cdmi-container
  - https://<offering>/containerB
  - Key Value
  - Key Value
  - ...

- **Domains** - cdmi-domain
  - https://<offering>/cdmi_domains
  - Key Value
  - Key Value
  - ...

- **DataObject1** - cdmi-object
  - https://<offering>/containerB/dataobject1
  - Key Value
  - Key Value
  - ...

- **DataObject2** - cdmi-object
  - https://<offering>/containerA/dataobject2
  - Key Value
  - Key Value
  - ...

- **Queue** - cdmi-queue
  - https://<offering>/containerA/queue
  - Key Value
  - Key Value
  - ...
All of these interfaces support some or all of this model. The key to retaining the simplicity of the cloud, however, is in the use of metadata to drive the underlying services so that users need not manage the services themselves.
Policies of the form *IF* `<condition>` *THEN* `<action>`

- Data System Metadata part of the `<condition>` side of Data Policies
- Data Services are invoked as part of the `<action>` side
- Standard Data System Metadata can be extended
- Additional Data Services can be added over time
- Offerings can offer any combination of data services for their customers
  - Discovered through CDMI Capabilities

CDMI query and queues can be used by add-in Data Services
- FP7 project VISION Cloud using this approach - “active storage”
- Java based “storelets” can be uploaded to the cloud and activated
Domains of Resource Management

**Information Resource Domain**
Services understand the semantics of the content in context

**Data Resource Domain**
The content is opaque to the Services and without context

**Storage Resource Domain**
The bits are contained by these Services

**Information Policies**
Data classified according to importance to organization

**Data Policies**
Data treated according to requirements, lifecycle

**Storage Policies**
Ensure correct and reliable operation
Being developed by Storage Networking Industry Association (SNIA), Long Term Retention (LTR), Technical Working Group (TWG)

An Analogy

- **Standard physical archival box**
  - Archivists gather together a group of related items and place them in a physical box container
  - The box is labeled with information about its content e.g., name and reference number, date, contents description, destroy date

- **SIRF is the digital equivalent**
  - Logical container for a set of (digital) preservation objects and a catalog
  - The SIRF catalog contains metadata related to the entire contents of the container as well as to the individual objects
  - SIRF standardizes the information in the catalog
A SIRF container includes:

- **A magic object**: identifies SIRF container and its version
- Numerous **preservation objects** that are immutable
- **A catalog** that is
  - Updatable
  - Contains metadata to make container and preservation objects portable into the future without external functions
Cloud Data Management Interface (CDMI) specifies a standard API for clouds.

CDMI API can be used to access the various preservation objects and the catalog object in a SIRF-compliant container.

Example:

- Assume we have a cloud container named "PatientContainer" that is SIRF-compliant:
  - each encounter is a preservation object
  - each image is a preservation object
  - the container has a catalog object

We can read the various preservation objects and the catalog object via CDMI REST API as follows:

```
GET <root URI>/PatientContainer/encounterJan2001
GET <root URI>/PatientContainer/chestImage
GET <root URI>/PatientContainer/catalog
```
Future feature candidates for CDMI

- **CDMI Profiles (may be done ancillary to specification)**
  - Simple Storage Management
  - Simple Storage

- **CDMI Extensions:**
  - Multi-part MIME GET/PUT
  - Key/Value Store support (i.e. NoSQL)
  - CDMI Jobs
  - Versioning
Unpacking OVF
- Treat as if it was “serialized”? i.e. point at OVF file “object” as source of a deserialize operation

Interpreting the envelope
- Find the files that would be CDMI exported containers
- “Mount” the .iso (for example) and export to the cloud computing environment (hypervisor instance)

Possible future
- Work with DMTF to scope CDMI Data System Metadata to include in OVF Manifest
Add a new capability: cdmi_ovf_support
- If present and true, OVF files can be the source of a container create

Create a container with an OVF file

application/ovf mime type CDMI Object
- JSON String
  - A data object serialized as an OVF file as specified in [OVF 1.1]

Result is a Container with exportable sub-containers and data objects corresponding to each file in the archive

Works with proposed OVF extension to CIMI
- VirtualSystemCollection maps to CIMI System class
- DiskSection of OVF maps to CIMI Volumes/CDMI exported containers
- **IaaS cloud** implements multiple cloud standards: CIMI, OVF, CDMI
- **Orchestration layer** implements the support
- **Client** uses each API for respective tasks: cloud computing, cloud storage, portability
Profiles are a means of “scoping down” CDMI for specific use cases

Clients can then depend on certain CDMI features being available if an implementation claims profile conformance

Profile consists of:
- Profile use case description (descriptive text and diagrams)
- List of required and optional CDMI capabilities that make up the profile
- “Programmers Guide” examples of using the profile from a client’s perspective (to encourage client side adoption of the profile as well)

Profiles intended to drive conformance programs
Many storage vendors have products that are sold with private management interfaces for the storage administrator.

They are looking for a self service RESTful API they can easily add to their products for cloud storage provisioning and data management.

Cloud Storage Administrator sets up pools of storage of different types: i.e. Gold, Silver, Bronze.

- Each with different capabilities exposed through CDMI.

Cloud Storage User picks a storage type and creates Containers, exporting them via iSCSI, NFS, CIFS, etc.

- Charge back is possible to the various departments for their use.
Many storage vendors are looking for a RESTful object storage data path to enable (alongside existing interfaces such as NFS and iSCSI).

CDMI can be used as such a data path.

The Simple Storage Profile describes using CDMI for the subset of capabilities that match the typical data path offering:
- i.e. Amazon S3, Rackspace Cloud Files, Nirvanix, etc.

CDMI allows interoperability without having to reverse engineer a de facto (under specified) API.

Implementations can be done from the profile document itself with only occasionally referring to the CDMI spec itself.
How do you know what your cloud can do?
Cloud Storage

Data System Metadata Capabilities

- **cdmi_data_redundancy**
  - If present, this capability specifies the maximum number of redundancy copies that may be specified. If absent, redundancy copies specified shall be ignored.

- **cdmi_infrastructure_redundancy**
  - If present, this capability specifies the maximum number of infrastructure redundancy copies that may be specified. If absent, infrastructure redundancy copies specified shall be ignored.

- **cdmi_data Dispersion**
  - If present and "true", indicates that the cloud storage system shall disperse data. If absent, redundancy copies specified shall be ignored.

- **cdmi_data_retention**
  - If present and "true", indicates that the cloud storage system shall support retention.
cdmi_data_autodelete
- If present and "true", indicates that the cloud storage system shall support the autodeletion of objects when retention ends.

cdmi_data_holds
- If present and "true", indicates that the cloud storage system shall support placing holds on objects. When a CDMI implementation supports holds for the purpose of making data immutable, the system-wide capability of "cdmi_security_immutability" specified in Section 12.1.1, "Cloud Storage System-Wide Capabilities" shall be set to "true". Otherwise, it shall not be present, indicating that there is no support for data immutability.
**cdmi_data_holds**

- If present and "true", indicates that the cloud storage system shall support placing holds on objects. When a CDMI implementation supports holds for the purpose of making data immutable, the system-wide capability of “cdmi_security_immutability” specified in 12.1.1, "Cloud Storage System-Wide Capabilities" shall be set to “true”. Otherwise, it shall not be present, indicating that there is no support for data immutability.

**cdmi_encryption**

- If present, this capability lists the encryption algorithms and key lengths supported. If absent, objects shall not be encrypted. When a CDMI implementation supports at-rest encryption, the system-wide capability of “cdmi_security_encryption” specified in "Cloud Storage System-Wide Capabilities" shall be set to “true”. Otherwise, it shall not be present, indicating that there is no support for at-rest encryption.
Data System Metadata Capabilities

- **cdmi_value_hash**
  - This metadata is used to indicate if the object data is to be hashed and indicates the desired hash algorithm and length. Supported algorithm/length values are provided by the cdmi_value_hash capability.

- **cdmi_latency**
  - If present and "true", indicates that the cloud storage system shall tier data based on desired latency. If absent, the max latency specified shall be ignored.

- **cdmi_throughput**
  - If present and "true", indicates that the cloud storage system shall tier data based on desired throughput. If absent, the max throughput specified shall be ignored.
Data System Metadata Capabilities

- **cdmi_sanitization_method**
  - If present, this capability lists the sanitization methods supported. When a CDMI implementation supports sanitization, the system-wide capability of “cdmi_security_sanitization” specified in Table 10 of Section 12.1.1, "Cloud Storage System-Wide Capabilities" shall be set to “true”. Otherwise, it shall not be present, indicating that there is no sanitization support.

- **cdmi_RPO**
  - If present and "true", indicates that the cloud storage system shall manage data to achieve a specified RPO. If absent, the RPO specified shall be ignored.

- **cdmi_RTO**
  - If present and "true", indicates that the cloud storage system shall manage data to achieve a specified RTO. If absent, the RTO specified shall be ignored.
How does CDMI fit into a storage cloud?

- **Large Scale out Cloud**
  - Deployed as a horizontal set of parallel filesystem clients with requests balanced across them
  - Storage is implemented by a set of data servers with a common metadata server
One Web Site to Remember: http://snia.org/cloud

Online Version of CDMI: http://cdmi.sniacloud.com

Offline questions: mark.carlson@oracle.com