A Fedora Akubra Storage Plugin for the Dell DX Object Storage Platform

Thomas Habing, Howard Ding, William Ingram, Robert Ferrer

Introduction

The Library at the University of Illinois at Urbana-Champaign is developing a digital preservation archive using the Fedora Commons Repository Software and the Hydra Framework. A major consideration in the development of this archive was selection of a storage platform. Although we had settled on the software platform early in the project, there was significant uncertainty with regard to the storage layer.

A primary requirement was that all data be replicated with at least three copies in disperse locations. Another imperative was the ability to monitor the integrity of all copies using fixity validation. We needed multiple, continuously validated copies in different locations to ensure the safety of the files. Cost, scalability, and the long-term viability of the storage solution were also major concerns. Simplicity of implementation and deployment and simplicity of ongoing management was also a priority. We calculated an initial space requirement of about fifty terabytes with an estimated growth of about seven terabytes per year. We also anticipated a growing need to store and preserve media files in the several hundred gigabyte range. We were also looking at the possibility of opening submission into the repository to the campus at large, not just the library, in which case scalability would become an even greater concern.

Our original plan was to build the storage system ourselves using the traditional file-system approach. We leaned toward a JBOD solution (Just a Bunch of Disks) with two replicated storage clusters running on commodity hardware, one cluster in each of two library buildings on campus. We could use rsync to synchronize the two clusters, and we investigated the possibility of using a public cloud storage provider such as DuraCloud to maintain a third copy of the data. We could also employ ACE (Auditing Control Environment) to continuously validate the integrity of the files across the two storage clusters. We also investigated the possibility of creating a customized multiplexing Fedora Akubra storage plugin, possibly implementing some variation of the PairTree algorithm for dispersing blobs across our JBOD. Our roll-your-own approach, while certainly workable, was not without its risks, not least of which was the size of our small team with its limited resources and immediate need to fill.

We also considered the cloud-only solutions, like Chronopolis and DuraCloud. While we were willing to rely on the public cloud for backup and replication, we were dissuaded by security and trust issues, and by our requirement that data replication always span multiple locations. We were further dissuaded by the current cost of cloud storage, which is significantly higher than what it would take to house the data ourselves.

We were ready to purchase the hardware and begin building our storage system when we were introduced to the area of Object Storage. A handful of storage vendors and open source organizations are now developing storage platforms that differ from the traditional file storage model in that data is stored in addressable objects made up of encapsulated bitstreams. Such examples are EMC's Centera, Rackspace's Cloud Files, and the open source OpenStack Object Storage. Dell Computers, in partnership with Caringo software, have also entered the object storage arena with their DX Object Storage Platform, which we ended up buying. Out of the box, the DX seemed to meet nearly all of our storage requirements, meaning that we would have to build less ourselves or purchase costly add-ons to more traditional storage solutions. Having decided on the software layer for our repository, Fedora Commons, our primary development task would be to customize Fedora to manage objects on the DX.

What is the DX Object Storage Platform

The Dell DX Object Storage Platform is a hardware and software bundle composed of Dell commodity X86 servers integrated with the CAStor software from Caringo. A baseline storage cluster consists of a single Cluster Service Node (CSN) and at least two Storage Nodes (SNs). The CSN provides several key functions for the storage cluster, but it primarily serves as a network PXE boot (Pre-Execution Environment) and configuration server for the storage nodes. All storage nodes added to the cluster PXE boot and acquire their configuration from the CSN. This allows for easy provisioning of additional SNs. It also means that the SNs do not require an operating system of their own, which allows all the disks on an SN to be used for storage. The CSN also operates as the management console for the cluster and includes SNMP for status monitoring. Finally, the CSN hosts the Content Router (CR), which employs a publisher/subscriber model where objects can be published based on metadata. This is used primarily to enable asynchronous replication between remote storage clusters, but it can also be used for third-party content processing applications like virus scan, indexing, or format migrations.

The DX supports configurable replication both within a single cluster and across a WAN. At object creation, one can specify the required number of copies (at least two) and specify *life points* used to control how many copies are required at different points in the object's life cycle. Continuous checksum- and cardinality-based integrity checks are performed on all objects. If any degradation is discovered, new copies are spawned automatically by the DX's *health processor* as needed to maintain the number of copies specified by the object's current life point. When more storage is needed, it is easily provisioned by adding additional SNs; they PXE boot automatically from the CSN and immediately begin accepting data. If an SN fails and needs to be removed, the health processor will promptly begin spawning additional copies to make up for the loss. All interaction with the storage is made through a RESTful API which maps the HTTP verbs POST, GET, PUT, and DELETE to the CRUD Create, Retrieve, Update, and Destroy storage functions. The system provides for custom metadata attributes on all objects, which are serialized as HTTP header fields. Objects can be created as mutable or immutable, and are assigned a UUID by the system or given a name by the storing application.

DX Deployment at UIUC

We deployed the DX storage cluster across two buildings on campus such that every storage object was replicated at both locations, with a third copy that *floated* between the sub-clusters depending on resource availability. The University of Illinois Main Library and the Grainger Engineering Library are connected by a 10 gig network and are about one quarter mile apart, so we enabled the sub-clustering feature of the DX, which guarantees that content is replicated between the sub-clusters. Generally, for this configuration to work the two sub-clusters must be on the same L2 subnet. However, our campus network policy prohibits the use of an L2 network across building nodes. No two nodes can share a single network VLAN. So we improvised. Working with our campus Network Services and Dell we designed and implemented a custom configuration such that a CSN and two SNs were deployed in each building. The CSN at Grainger acts as the primary manager of the entire cluster. The second CSN serves merely as a PXE boot server for the SN nodes at the Main Library. Using the second CSN this way is more economical than licensing a complete second cluster for communicating across the WAN via the content router, which initially appeared to be our only alternative. Custom configuration of campus network routers was required to allow multicast network packets to cross subnets. Storage nodes across subnets multicast using the Rendezvous Point protocol to prevent network flooding. The routers were set for Jumbo Frames to maximize throughput. This way we were able to configure a single cluster with two sub-clusters across an L3 network, keeping with University policy while maintaining optimal performance. Expansion of the system is possible with the acquisition of additional storage nodes. To maintain the symmetry of the sub-clusters we will add two SNs at a time, one per sub-cluster, but a single SN can be added to either sub-cluster as needed. Any SN can respond to a request for any object, so to facilitate load balancing and fault-tolerance a single canonical domain name is bound to all SNs in the cluster and round-robin DNS with a short time-to-live (TTL) is used to ensure that requests are equally distributed between the nodes.

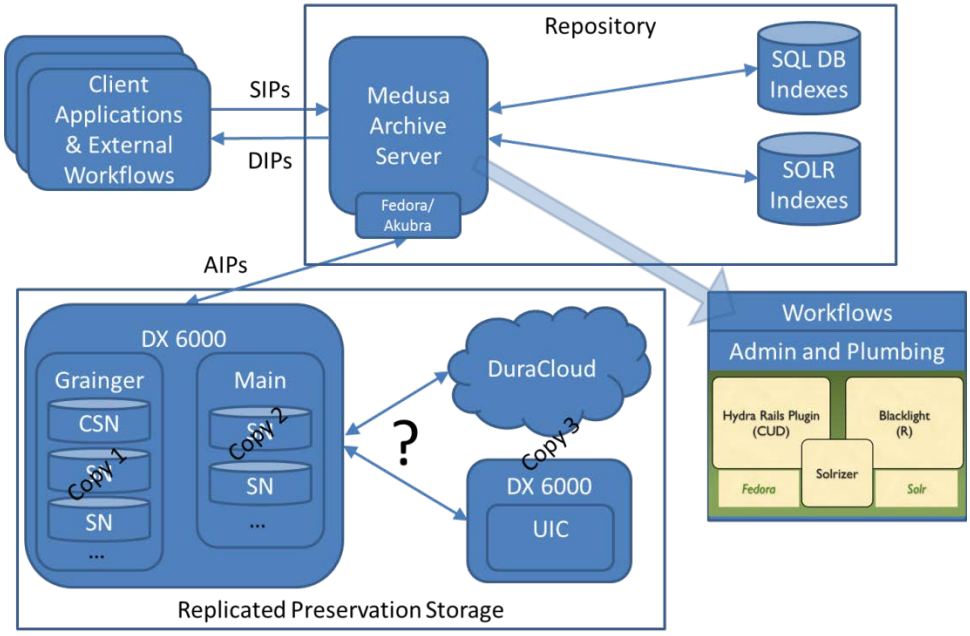

Figure 1 High-level Architecture

Akubra DX Plugin Details

Akubra Low Level Storage is the pluggable storage interface that was introduced in Fedora version 3.2. The default Akubra plugin used by Fedora stores bitstreams on the file system according to some simple rules and configuration settings. However, Akubra was designed to be easily adapted to any storage subsystem, and this enabled us to customize the interface to work with the DX.

The base interface to the DX Storage Platform uses a subset of the HTTP 1.1 protocol called the Simple Content Storage Protocol (SCSP). SCSP maps the various storage CRUD actions to HTTP verbs and defines how URL query string parameters and HTTP headers should be utilized by the system. Clients can directly use the HTTP protocol and SCSP to interface with the DX storage. For example, retrieval is an HTTP GET request: http://libstor.grainger.illinois.edu/open/info:fedora/MEDUSA:f7236f68-5c00-4837-a7a0-af387823de45-4?domain=medusa.grainger.illinois.edu. However, there are also SCSPClient APIs available for several programming languages including Java.

By inheriting from Akubra abstract classes and using SCSPClient classes and methods, we were able to implement an Akubra plugin for the DX storage platform. Akubra specifies a very restricted set of operations for bitstreams (Blobs), and implementation was largely a matter of mapping these operations to services provided by the DX API while converting any DX errors into appropriate Akubra counterparts so that Akubra's client (in this case Fedora) could take the correct action on failure. We also have used (or abused) the Akubra "hint" facility to allow us to create DX metadata headers. Our current implementation is fully functional but there are some known improvements still to be made.

Future Work

There are several enhancements to our DX-based storage system that we plan to explore as the project moves forward. First among these is better utilization of the DX's metadata capabilities. The DX supports custom metadata headers which can be specified as `x-*-meta-*`. We are currently utilizing this capability to attach the Fedora stream id and also the repository name to each object, such as this:

```
x-fedora-meta-repository-name: development-test-repo
x-fedora-meta-stream-id: info:fedora/MEDUSA:f7236f68-5c00-4837-a7a0-af387823de45-4
```

However, we are also exploring how we can expose additional metadata properties to the DX. For example, it might be useful to add the Global Digital Format Registry (GDFR) identifiers to the DX metadata. This would allow us to use the Content Router to enumerate all objects of a specific format to perform conversion actions. It might also be useful to expose Dublin Core (DC) metadata to the DX, such as these:

```
x-dc-meta-title: Map of Tripoli
x-dc-meta-isPartOf: info:fedora/MEDUSA:f7236f68-5c00-4837-a7a0-af387823de45-4
x-fedora-meta-gdfr-id: info:gdfr/f/formatid
```

Similarly we are also investigating how best to use the Content Router to reconstitute a Fedora repository from just the data bitstreams. We feel this is critical functionality for preservation, and we want to ensure that the `fedora-rebuild` command will work with the DX. The Fedora code for rebuilding relies on enumerating all objects through Akubra and then accessing them and any of their managed datastreams (again, through Akubra). For filesystems this is easy – the streams are stored in designated directories with a standard naming scheme. By using the Content Router to retrieve all repository objects and their pids, which we can do because of our metadata headers, we believe that we will be able to rebuild the Fedora repository with only what is stored on the DX.

References

Caringo, Inc.. Dell DX Object Storage Platform: DX Object Storage Application Guide, Version 5.0.1. Dell, 2011. http://support.dell.com/support/edocs/systems/IDM_com/DX_SD/OSAG.pdf

Caringo, Inc.. Dell DX Object Storage Platform: Software Development Kit (SDK) Overview, Version 5.5. Dell, 2012.

Carpentier, Paul. Traditional storage models disrupted! Storage Expo Brussels, March 24, 2010. http://www.caringo.com/downloads/presentations/StorageExpo_Brussels_Keynote.pdf

Dern, Daniel P. "Caringo Object Storage Software Puts SMBs On Cloud" InformationWeek, October 26, 2010. http://www.informationweek.com/news/smb/services/228000077

Duracloud. http://www.duracloud.org/

Farmer, Jacob. Making a Case for Out-of-Band File System Middleware: For Digital Asset Management. Cambridge Computer, 2010. http://www.digitalpreservation.gov/meetings/documents/othermeetings/23_farmer.pdf

Kunze, J. Pairtrees for Collection Storage https://confluence.ucop.edu/display/Curation/PairTree

McClure, Terri. Archive TCO: Five-Year TCO Comparing Dell DX to Tape and NAS for Long Term Archive. Enterprise Strategy Group, 2011. http://www.dell.com/downloads/global/products/pvaul/en/dell-dx-tco.pdf

Minor, D., D. Sutton, A. Kozbial, M. Burek, M. Smorul. Chronopolis Digital Preservation Network. 5th International Digital Curation Conference, December 2009. https://chronopolis.sdsc.edu/publications/assets/docs/chronopolis_dcc_revised.pdf

Kerner, Sean Michael. "Object-Based Storage Debated at Interop." InfoStor, May 11, 2011. http://www.infostor.com/storage-management/object-based-storage-debated-at-interop.html

Smorul, M., Song, S., and JaJa, J. An Implementation of the Audit Control Environment(ACE) to Support the Long Term Integrity of Digital Archives in DigCCurr 2009. 2009: University of North Carolina. https://wiki.umiacs.umd.edu/adapt/images/5/5b/DigCCurr2009_060909.pdf

Wilper, Chris. Akubra Project. https://wiki.duraspace.org/display/AKUBRA/Akubra+Project