From the Desktop to the Cloud: Leveraging Hybrid Storage Architectures in your Repository

David Tarrant, Tim Brody, and Leslie Carr

School of Electronics and Computer Science, University of Southampton, Southampton, UK,
`dct05r,tdb2,lac @ecs.soton.ac.uk`

Abstract. Repositories collect and manage data holdings using a storage device. Mainly this has been a local file system, but recently attempts have been made at using open storage products and cloud storage solutions, such as Sun's Honeycomb and Amazon S3 respectively. Each of these solutions has their own pros and cons but There are advantages in adopting a hybrid model for repository storage, combining the relative strengths of each one in a policy-determined model. In this paper we present an implementation of a repository storage layer which can dynamically handle and manage a hybrid storage system.

Repositories collect and manage data holdings using a storage service. The ever increasing set of demands applied upon a repository means that ingesting, managing and provisioning access to vast amounts of data using only local disk storage is a difficult task. While there are storage options which can help in these tasks, managing these dynamically is not something which can be done in currently available repository software. Attempts have been made to combine the various repository softwares with various storage platforms such as Amazon S3 and the Honeycomb (STK5800) from Sun Microsystems, however these have been focused on utilising these platforms as a replacement for local disk storage.

With many storage options now becoming available we realised the advantages to the repository community of adopting a hybrid platform which is suited to the needs of your repository or institution. Table 1 provides a brief overview of various storage solutions which can be utilised, giving the pros and cons of each.

This hybrid model is controlled with a storage policy, enabling a repository to utilise many types of storage dependent on the needs of that repository or institution. We list here a few of the use cases where a hybrid model would aid the repository. Each of these solutions, whilst stated as a singular use case, can be combined with each other to suit the specific needs of each repository.

– A local copy for security and long term preservation: As per the current repository model, this copy provides the assurance that if other services were to be lost, that there would still be a version in existence of that object.

Type	Pros	Cons
Local Disk	No local bandwidth costs	Hard to expand
	Locally Managed	High overheads cost
		Requires space and cooling
		Tied closely to the software
Local Archival Storage	Specialist	Expensive to purchase
	Locally Managed	Space and running costs
	Expandable	
Cloud Storage	Scalable	Externally controlled
	Known Costings	Unclear retention policy
	Re-Useable (using simple APIs)	Bandwidth costs

Table 1. Storage Types: Pros and Cons

– Cloud Storage: By utilising cloud storage services such as Amazon S3 and Cloudfront, enables construction of a highly distributed, high bandwidth, scalable storage solution which is distributed over many continents. Hosting repository objects in the cloud also enables low bandwidth institutions to host large scale repositories, protect against sudden high loads.
– Repository Mirroring: A set of institutions could agree to mirror each others content in order to protect against system failure and natural dissasters affecting the source repository.
– Utilise specialised archival and preservation service providers: By plugging your storage layer directly into an archival or preservation system you could further guarantee the longevity of your objects. Such services could range from simple tape backup providers to more complex providers who also analyse the contents of your repository and provide risk analysis reports.

In a fully hybrid model, each stage of the repository object lifecycle could also utilise different storage platforms suited to the particular needs of that stage. As a simplistic example, in a system where your default policy is to ingest everything before the appraisal and selection stage, you could use low availability storage for the ingest, transferring the accepted contents into a larger, higher bandwidth system ready for end users to download. This is just one of many policy decisions that needs to be available to the repository manger, other decisions could be made based upon the type, category and submitter of the content.

In order to implement and enable this hybrid storage platform, the EPrints platform was chosen to be the base software on which these services can be developed. EPrints was just one of the platforms which had already seen developments to enable use of technologies such as Sun's Honeycomb server, however it was not possible to utilise a hybrid storage solution.

By abstracting the way in which EPrints handles the storage of each file, a storage controller was developed that now handles the storage of each individual object. A simple API was devised against which plug-ins can be written to connect the storage controller with the various storage platforms. With the storage controller acting as a layer between EPrints and the storage plug-ins we can impose a set of simple rules in the storage controller that decide which plug-in is used for each storage operation. Figure 1 depicts the changes made as well as a simple ruleset which stores files in different locations.

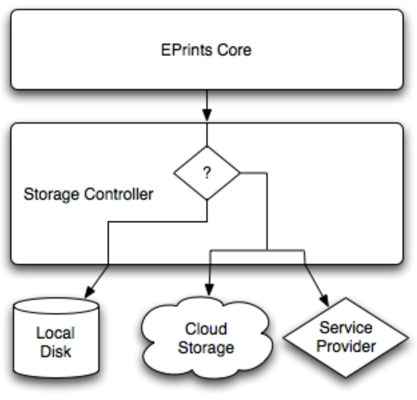

Fig. 1. Architectural Overview of System

The ruleset is written using the built in XML-based EPrints scripting language. Using this enables access to the full EPrints model and data about not only the file you are storing but also its properties and parent relations to the EPrint object itself. The following code block shows a simplistic example where we are storing all non-volatile files using the Amazon S3 service; thus all thumbnails, previews and text indexes are stored locally with the actual object being stored by S3.

```
<epc:choose>
      <epc:when test="datasetid = 'document'">
          <epc:choose>
              <epc:when test="$parent{relation_type} = 'isVolatileVersionOf'">
                  <plugin name="Local"/>
              </epc:when>
              <epc:otherwise>
                  <plugin name="AmazonS3"/>
              </epc:otherwise>
          </epc:choose>
      </epc:when>
      <epc:otherwise>
          <plugin name="Local"/>
      </epc:otherwise>
  </epc:choose>
```

At the time of writing three storage plug-ins are available for the storage controller, the local storage plug-in that also supports the legacy local disk layout, a Honeycomb plug-in for the Sun STK5800 server and an Amazon S3/Cloudfront plug-in.

In order to conform to more advanced web architecture demands the storage controller allows direct delivery of the object to the user from the storage platform. For example, each object in the Amazon Cloudfront service can be accessed via its own web location (URL), which is provided by the storage layer to the end user. This enables advanced features such as load-balancing and global locationing to be provided to the repository by external services, meaning less load on your repository system.

Being able to make use of hybrid storage platforms is inadequate without being able to manage and migrate data between them, in order to mitigate against service withdrawl. Within the EPrints platform modifications were made such that as each storage operation is performed, a note is made in the metadata that details the location of the object and the platform used to store it. Doing this also allows many copies of each object to be stored and tracked on different platforms. Migration can be done by executing a batch process that reloacates objects from the old storage platform into a new one, as defined by the hybrid storage policy.

The EPrints Storage Controller is proposed as an important step to enable the creation of flexible, scalable, high bandwidth distributed repositories. On top of this you gain the ability to act directly with service providers who provide added value for your repository such as added security, resilience or preservation capabilities. We also envisage that using the storage controller will enable repositories to harness the power of cloud computing for performing operations on their objects. This will allow the repository to integrate with emerging network, storage and cloud services.