TwoStore : An Akubra-based extensible storage architecture for Fedora

Adam Soroka and Tim Stevens

University of Virginia – Online Library Environment

How it STARTS…

"Hey, IT…can I have **10 Terabytes**?
In a **single volume**?
Oh, and my repo needs **one mount point**.
Please."

According to Moore's Law, the number of transistors on a chip roughly doubles every two years. As a result, the scale gets smaller and the chips run faster. While this statement applies to processors, **the same concept should be applied (and anticipated) when planning your digital repository storage needs**.

Your actual factor will depend on your organization, but **your storage needs will never get smaller**. Without proper planning and infrastructure will only run slower.

Initial Challenges:

- Many organizations simply **do not have the available hardware and management resources** to provide the necessary space.
- For those with the resources, even today, **providing massive storage in single volumes is difficult or impossible**.
- **The larger the volume, the longer it takes to copy and backup or to restore** in the event of a catastrophic disk failure.

How it gets WORSE…

"Hey, IT…I need **MORE space**.
Perhaps **another 10 Terabytes**?
It still has to have a **single mount point**.
Pretty Please."

Congratulations! It looks like you have a very successful repository -- so popular, in fact, that **you are about to run out of storage space**.

You will encounter **a never ending cycle of data migrations** that frustrate IT departments **and put your data at serious risk of corruption**.

New Challenges:

- A **complex configuration of symlinks** could be used for the volumes to appear as unified storage. *A hack at best*, but the only answer for many.
- With some repositories, **the volumes themselves contain metadata** required to differentiate the volumes and locate the files.
- If volumes contain their own metadata, **the disks cannot be restored independently of each other** and if one fails **the entire structure may be rendered unusable**.

How it gets SOLVED…

TwoStore operates over multiple filesystems and *is agnostic* as to how those filesystems are assembled and mounted to repository servers.

- A new Akubra **IDMapper** translates **external URIs** into a simple hash value consisting of **a binary string and an URL-encoded filename**, by a fast hashing algorithm implemented in only a few lines of Java.
- A new Akubra **BlobStore/BlobStoreConnection** pair multiplexes together two subsidiary BlobStores based on the beginning of the binary string.
- Concrete BlobStores **are connected** through the aforementioned multiplexing BlobStore.
- A utility **separates any FSBlobStore** using this type of internal URI **into two other FSBlobStores**, each of which contains half of the original Blobs, **using the same decision that the multiplexing BlobStore** uses in operation.

Together they are used to form a binary tree that is **quickly traversed for storage and retrieval**.

This construction offers **a number of advantages**:

- **New storage can be added quickly** by using our utility to migrate a pre-existing BlobStore into a multiplexing store with the original BlobStore as one child and the new BlobStore as the other. The migration can occur while a repository is operating.
- **No special metadata is included** inside the concrete BlobStores, which simplifies management operations. Any filesystem that supports POSIX semantics will do.
- **The total amount of storage available to a repository is 2^32 BlobStores over the quantity of storage available from the concrete BlobStores**. This is an extraordinarily large amount for such a simple plan which uses no complicated grid machinery.
- As new BlobStores are introduced, more of **the storage/retrieval decision tree is traversed in memory**, which provides good scaling characteristics.

The TwoStore Process