Migrating our Hydra Repository from Fedora 3 to Fedora 4

Jim Coble
Duke University Libraries
Duke Digital Repository

- Launched Summer 2013
- Preservation as a primary concern
- 336,000+ objects
  - 540 Collections
  - 78,000+ Items
  - 256,000+ Components (digital content files)
  - 9+ TB of data
- Two Hydra applications
  - Staff interface
  - Public interface
Migration Sub-Projects

- **Application Migration (Staff and Public)**
  - ActiveFedora 7 -> ActiveFedora 9

- **Content Migration**
  - Fedora 3 -> Fedora 4

- **System Architecture Migration**
  - Single server -> Separate Hydra application & Fedora/Solr servers
  - Scientific Linux 6 -> RHEL 7
  - Change in campus storage provider
Migration Team

● Core Models and Staff Application; Content Migration and Verification
  ○ David Chandek-Stark, Developer
  ○ Jim Coble, Developer
  ○ Jim Tuttle, Head, Digital Repository Services

● Public Application
  ○ Sean Aery, Developer
  ○ Cory Lown, Developer

● Consultant
  ○ Adam Wead, Hydra and Fedora-Migrate developer
Migration Decisions

- Retain existing content models
  - Not migrate to PCDM at this time
- Migrate only current version of objects
  - Few digital content objects had more than one version
- Transition RDF datastreams to object properties
  - Administrative metadata
  - Descriptive metadata
Fedora 3 to 4 Mismatches

- External datastreams
  - ⇒ Move file location to object property

- Roles represented using nested RDF structure with blank nodes
  - ⇒ Change role representation during migration and store as object property

- Checksum algorithm
  - SHA-256 (our Fedora 3 choice) and SHA-1 (Fedora 4)
  - Primarily a concern vis-a-vis verifying the migrated content
  - ⇒ Calculate source SHA-1 on the fly during migration and compare to target SHA-1
Fedora-Migrate Gem

- [https://github.com/projecthydra-labs/fedora-migrate](https://github.com/projecthydra-labs/fedora-migrate)
- Most customizations done via use of Fedora-Migrate callbacks
  - Source and target object integrity checks
  - Handling external datastreams, RDF blank nodes, SHA-1 checksums, original filenames
  - Merging administrative and descriptive metadata RDF datastreams
- Some method overrides
  - To let Fedora 4 generate ID’s for migrated objects
  - To use Fedora 3 PID’s to find Fedora 4 objects
  - To handle files larger than 2 GB
- Created Resque jobs for migration phases
  - Object migration
  - Relationship migration
  - Structural metadata migration
Migration Workflow

● Lock collection in Fedora 3

● Migrate collection
  ○ Objects, properties, datastreams (attached files)
  ○ Relationships
  ○ Structural metadata

● Validate migrated collection (independent of migration process)

● Version collection objects in Fedora 4

● Unlock collection in Fedora 4
## Migration Timeline

<table>
<thead>
<tr>
<th>Date</th>
<th>Event Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fall 2015</td>
<td>Engaged Adam Wead as consultant for project</td>
</tr>
<tr>
<td>October 2015</td>
<td>Began work on Fedora 4 version of application code</td>
</tr>
<tr>
<td>May 2016</td>
<td>Implemented Fedora 4 version of staff application in production</td>
</tr>
<tr>
<td>May 23, 2016</td>
<td>Began production object migration</td>
</tr>
<tr>
<td>July 1, 2016</td>
<td>Completed production object migration and verification</td>
</tr>
<tr>
<td>July 2016</td>
<td>Version(?) and unlock migrated objects</td>
</tr>
<tr>
<td>July 2016</td>
<td>Implement Fedora 4 version of public application in production</td>
</tr>
</tbody>
</table>
Migration Experience

- Set up completely new system (software, servers, OS, storage) and pointed a fire hose at it -- might not have been the best plan :-) 
- Successfully migrated 324,266 Fedora objects
- Bumps in the road
  - Early testing (using LevelDB)
    - Tomcat crash left corrupt repository object that we could not remove ⇒ Decided to wait for Fedora 4.5.1 and MySQL support
  - Production migration
    - Auto-versioning led to version creation failures ⇒ Turn off auto-versioning for migration
    - Locks held by long-running, frequent Infinispan DELETE query led to migration failures ⇒ Increase innodb_lock_wait_timeout (but there’s a better solution)
    - Problem versioning certain migrated objects ⇒ Still working on this one
Tips

- Have plenty of RAM
  - We ended up increasing all servers to 32GB
- If using MySQL or PostgreSQL, index the ‘version’ (timestamp) column of the Infinispan database (ispn_entry_FedoraRepository)
  - Solution (we think) to long-running Infinispan DELETE query
- Be prepared for issues to arise as size of Fedora 4 repository grows
  - They did for us (e.g., long-running Infinispan DELETE query)
- Verify migrated objects if you can
  - For us, uncovered a few issues not caught during migration itself
- Expect it to take a while
  - Our migration took about 5 weeks
Contact Info

Jim Coble
Duke University Libraries

jim.coble@duke.edu