# Bridging the Gap Between Real World Repositories and Scalable Preservation Environments*

Bolette Ammitzbøll Jurik
State and University Library
Victor Albecks Vej 1
DK-8000 Aarhus C, Denmark
baj@statsbiblioteket.dk

Asger Askov Blekinge
State and University Library
Victor Albecks Vej 1
DK-8000 Aarhus C, Denmark
abr@statsbiblioteket.dk

Rune Bruun Ferneke-Nielsen
State and University Library
Victor Albecks Vej 1
DK-8000 Aarhus C, Denmark
rbf@statsbiblioteket.dk

Per Møldrup-Dalum
State and University Library
Victor Albecks Vej 1
DK-8000 Aarhus C, Denmark
pmd@statsbiblioteket.dk

## ABSTRACT

Integrating large scale processing environments, such as Hadoop, with traditional repository systems, such as Fedora Commons 3, have long proved a daunting task. In this paper we show how this integration can be achieved using software developed in the SCAPE project. The SCAPE integration is based on four steps: retrieving the metadata records from the repository, reading the records and their references to data files, updating the records, and storing them back in the repository. This allows full use of the Hadoop system for massively distributed processing without causing excessive load on the repository.

We present a proof of concept integration based on repository systems at the Danish State and University Library and the Hadoop execution environment. As a sample collection we use data from the Newspaper Digitisation Project, a collection of more than 30 million JP2 images. The use case is to perform feature extraction and validation of the JP2 images. The validation is done against an institutional preservation policy expressed in the machine readable SCAPE Control Policy vocabulary. The feature extraction will be done using the Jpylyzer tool. We perform an experiment with various-sized sets of JP2 images, to test the scalability and correctness of the solution.

We show that it is both possible and beneficial to use this approach when having to perform preservation actions on massive collections stored in traditional digital repositories.

*This work was partially supported by the SCAPE Project. The SCAPE project is co-funded by the European Union under FP7 ICT-2009.4.1 (Grant Agreement number 270137).*

978-1-4799-5569-5/14/$31.00 

## Categories and Subject Descriptors

E.2 [**Data**]: DATA STORAGE REPRESENTATIONS—*Object representation*; H.3.4 [**Information Systems**]: INFORMATION STORAGE AND RETRIEVAL—*Systems and Software, Performance evaluation (efficiency and effectiveness)*; H.3.7 [**Information Systems**]: INFORMATION STORAGE AND RETRIEVAL—*Digital Libraries, Collection*

## General Terms

Design, Experimentation, Performance

## Keywords

Digital Preservation, Digital Repository, Preservation Action, Preservation Policies, Scalability, Integration, File Characterisation, JPEG 2000, Apache Hadoop

## 1. INTRODUCTION

In this paper we present a case study on integrating a real world digital object repository with a scalable preservation action processing environment. The repository is the Danish State and University Library's (SB) digital repository comprised of the Fedora Commons based DOMS repository [1] for metadata and the Bit Repository [2] for data. The processing environment is a local instantiation of the SCAPE [3] environment (see section 3) which is based on Apache Hadoop [4]. The preservation action is the validation of scanned newspaper pages against an institutional preservation policy.

### 1.1 Outline

In section 2 we begin by presenting the challenge of preserving large collections in general along with some examples. In section 3 we present the SCAPE environment for executing preservation actions and we describe the necessary SCAPE parts for such a task. In section 4 we describe the SB repository along with a deployed Apache Hadoop cluster.

To integrate the SCAPE environment with the SB digital repository we have developed the SCAPE DOMS Connector presented in section 5.

For the case study we in section 5.1 present an example of a preservation action involving an ongoing newspaper digitisation project that needs to characterise and validate more than 30 million image files.

In section 6 we conclude and present further work.

# 2. GENERAL DISCUSSION

The generic challenge is that, given a large collection in a digital repository along with preservation policies, these policies will require preservation actions to be performed on the collection.

## 2.1 Content Holders and Collections

Big libraries, museums and archives such as the SB hold large collections of digital material, and are charged with the preservation of these large digital collections. One example of such a large collection is the Danish Radio and TV broadcasts collection which in March 2014 stored more than 1.5PB audio and video files and grows by 1TB a day [5]. Another example is the 19th century British Library newspaper collection [6], which is around 80TB image files. Yet another example of archives holding massive amounts of data are web archives. The Danish net archive (netarchive.dk) holds in excess of 450TB of web archive content in uncompressed ARC format [7] and is growing by 100TB a year.

Common for these large content holders with preservation responsibilities is that they usually store their collections in digital repositories, and they usually define and enact preservation policies.

## 2.2 The Digital Repository

Digital preservation repositories are used to store collections of digital content, both data and metadata. They should support access to the repository content for designated user communities and enable the collection managers to monitor and perform preservation actions on the collections.

Of particular interest due to their adoption by the digital preservation community are the open source repository system Fedora Commons [8] version 3 and 4 and the Fedora Commons based systems of Islandora[9], Hydra[10], eSciDoc[11] and DOMS [1]. Other open source systems include DSpace [12], EPrints [13], dArceo[14] and Lily[15]. Commercial systems include Rosetta[16] and RODA[17].

## 2.3 Preservation Policies

A preservation policy can be described as a

> ... written statement authorised by the repository management that describes the approach to be taken by the repository for the preservation of objects accessioned into the repository. [18]

Examples of published preservation policies have been collected both on the SCAPE wiki [19] and on the Signal Digital Preservation blog [20]. A preservation policy at SB is e.g. that we prioritise migration over emulation.

## 2.4 Preservation Actions

Preservation actions are actions that should be performed on a collection to ensure that the data is preserved in accordance with the institutional preservation policies. This means that we need tools and environments that scale to the size of the collections and methods for these tools and environments to access and operate on the data in the collections i.e. the digital repository should support such operations.

An example of such a preservation action is the migration of the 19th century British Library newspaper images from TIFF to JP2 [21]. Another example of a preservation action is the characterisation of a web archive. Web archives often contain billions of files formatted using a very wide range of file formats including obsolete and forgotten formats. This makes characterisation of such an archive a tremendous challenge.

When the data size becomes as big as in the examples above the task of manipulating it becomes inherently hard. Data size in itself adds complexity to a system when it grows into what is popularly called Big Data. In the last decade since Dean and Ghemawat [22] published their paper on the MapReduce framework, the answer to most distributable problems has been MapReduce. In the world at large MapReduce has become synonymous with the Open Source Apache Hadoop system which is also what we will use when addressing these large scale preservation action challenges.

# 3. THE SCAPE PROJECT

Scalable Preservation Environments (SCAPE) [3] is an EU-funded FP7 project which is directed towards long term digital preservation of large-scale and heterogeneous collections of digital objects. It aims to develop scalable services for preservation planning and preservation actions on an open source platform. These services are based on a framework for automated, quality assured workflows, which are being elaborated and tested during the project. A policy-based preservation planning tool and an automated watch system will ensure a secure and targeted implementation of institutional preservation strategies.

As a SCAPE partner organisation, we use a number of SCAPE components, which will be described in the following sections.

## 3.1 The SCAPE platform

The SCAPE project has dedicated a sub-project to create a platform that supports the execution of large scale preservation actions and describes it thus

> The SCAPE Preservation Platform provides an extensible infrastructure for the execution of digital preservation processes on large volumes of data. A key challenge is the development of methods to integrate preservation tools and diverse data corpora with a massively parallel execution environment [23].

Prior to the implementation phase of the SCAPE project an investigation concluded that the massively parallel execution environment would be based on Apache Hadoop. Using Apache Hadoop poses certain demands and requirements on the existing infrastructure of an institution. These demands and requirements are both organisational and technical. Bringing the computing power to the data is not a trivial task in an existing organisational, political, and technical infrastructure.

## 3.2 The SCAPE Data Model

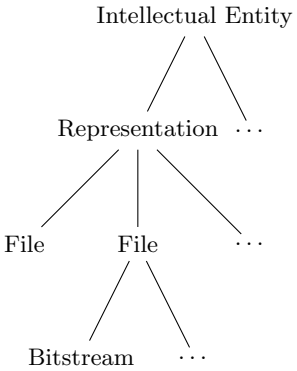


Figure 1: SCAPE Data Model

The SCAPE Data Model [24, 25] is based on the PREMIS [26] Data Dictionary for Preservation Metadata which is the international standard for metadata supporting the preservation of digital objects and ensuring their long-term usability. The SCAPE Data Model shown in Figure 1, is based on the four levels of objects from the PREMIS Data Dictionary; Intellectual Entity, Representation, File and Bitstream.

An Intellectual Entity is a set of content that is considered a single intellectual unit for purposes of management and description. An Intellectual Entity may have one or more Representations. Each Representation is the set of files, including structural metadata, needed for a complete and reasonable rendition of an Intellectual Entity. A Representation may have one or more File objects. Such a File object is a named and ordered sequence of bytes that is known by an operating system, and addressed as a URL. A file can be zero or more bytes and has a MIME type. Each File has zero or more Bitstreams that are contiguous or non-contiguous data within a file that has meaningful common properties for preservation purposes.

The SCAPE Data Model defines how SCAPE objects can be serialised into the Metadata Encoding and Transmission Standard (METS) [27] format. METS is a metadata standard for encoding descriptive, administrative, and structural metadata regarding objects within a digital library, expressed using the XML schema language of the World Wide Web Consortium.

## 3.3 The SCAPE Repository API

The SCAPE Repository API [28] is a uniform REST API for repository systems that allows a client to retrieve and submit objects expressed in the SCAPE Data Model. For this paper two methods are of particular importance. The first is the "Retrieve an Intellectual Entity", which retrieves an Intellectual Entity as a METS document. The second is "Update an Intellectual Entity", which takes a METS document and stores this back into the repository.

Using these two methods, a checkout/commit style interaction with the repository can be achieved, similar to the well known pattern used by software version control software:

1. A user retrieves one or more Intellectual Entities as METS documents.
2. The user updates the retrieved METS documents.
3. The user then submits the METS documents back to the repository.

To support this style of interaction, SCAPE is producing two client components, which integrates with the SCAPE Repository API. These are the Stager and the Loader, as seen in figure 2. The Stager is the checkout client, in that it allows one to retrieve Intellectual Entities as METS documents and store them on the execution platform. The Loader is the commit client, in that it reads Intellectual Entities as METS documents and updates the repository. It can be used both to create new objects in the repository and to update existing objects.

The SCAPE Repository API has already (March 2014) been implemented for the repository systems Fedora Commons 4 and Rosetta from Ex Libris, and, as will be described in this article, for DOMS.

## 3.4 The SCAPE Control Policy

The SCAPE Project has defined a model for preservation policies [29, 30] that supports organisations in creating their preservation policy documents. This model consists of three levels.

**Guidance Policy** Describes general long term preservation goals of the digital collection(s).

**Preservation Procedure Policy** Describes approaches the organisation will take in order to achieve the goals.

**Control Policy** Describes requirements for a specific collection, a specific preservation action, or a specific designated community. This level can be human readable, but should also be machine readable.

The SCAPE Control Policy Model provides a controlled vocabulary or set of terms and relationships that allow for the description of policies [29, 30]. Key entities described in the model are Content Sets and Objectives. A Content Set represents a collection of objects that is the focus of the policy. Objectives are the atomic building blocks of the policies. In general, an Objective will refer to a property along with a value for the property and a Modality. The Modality indicates whether or not the expected value is an absolute requirement or prohibition, expressed as MUST/MUST NOT/SHOULD/SHOULD NOT. In our case study we use control policy objectives expressing requirements about colour space and depth for a collection of JP2 image files (see section 5).

The policy model described above is available as a collection of OWL ontologies and SKOS vocabularies within the general namespace of http://purl.org/DP (for Digital Preservation).

## 3.5 Jpylyzer

Jpylyzer [31] is a validation and feature extraction application for JP2 images. JP2 is the image format defined by Part 1 of the JPEG 2000 image compression standard [32].

In the SCAPE Project, both the British Library, the National Library of the Netherlands and SB are using or plan to use the JP2 image format for long term preservation. There is, however, not much tool support, especially in the area of format validation. Jpylyzer was therefore developed in SCAPE to bridge this gap.

Jpylyzer is a Python command line program that produces XML output containing relevant properties of the image file. Jpylyzer will raise a flag for all images that do not adhere to the JP2 format specification. Further profile specific checks are not supported, but can be performed by doing analysis on the produced XML.

We did a layman's performance evaluation of Jpylyzer on a commodity laptop computer. Working sequentially on 10.000 JP2 files, Jpylyzer used 11 minutes and 20 seconds to validate the complete set of files. To check how much slower this was compared to just reading the files, we did fixity checking (which should be proportional to the I/O read speed) of the same files on the same machine, which took 7 minutes and 25 seconds.

# 4. THE IT SYSTEMS OF SB

At SB we have chosen to use one repository system for our metadata and another for the file data. This choice is based on the very different requirements for preservation of data and metadata. Data files should remain permanently immutable whereas metadata does not have such a requirement as we expect it to change over time when we discover mistakes, get better tools, or perform other preservation actions. The data files can, if succesfully migrated, be removed, but otherwise they should never be changed. The metadata files should instead be versioned, so that any changes can be tracked.

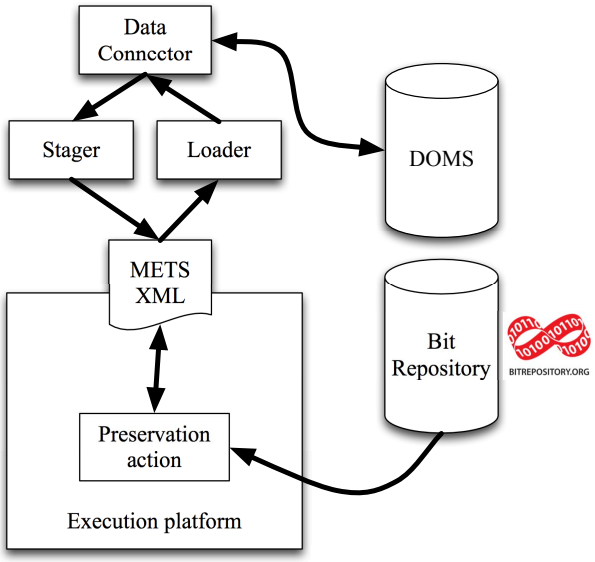


Figure 2: The Data Connector retrieves objects from DOMS. The Stager checks out objects as METS objects. The Preservation Action streams file content from the Bit Repository, works on it and updates the METS objects. The Loader commits the objects to the Data Connector which then updates DOMS.

## 4.1 DOMS

DOMS [1] is the repository system used for metadata at SB. It is based on Fedora Commons 3 [8], and uses the Enhanced Content Models extension [33].

Unlike many other repository systems, DOMS does not enforce a specific data model. Rather, each collection defines its own data model. These are expressed in the language of the Enhanced Content Models, allowing software connected to DOMS to adapt itself to the collection specific data model.

## 4.2 Bit Repository

Our data files are stored in the National Bit Repository [2] on multiple server systems, separated geographically and organisationally. To access the data, we request a set of files from the system. When the request has been approved it will be made available on a network share, connected to our processing cluster. Approving a request may include organisational processes as data can be stored on various technologies both on- and off-line.

## 4.3 Execution platform

The IT strategy at SB is to employ hardware virtualisation on enterprise servers for computation and network attached storage (NAS) for general data storage. For the work presented here we decided that we did not want to deploy a traditional Hadoop cluster. Instead we decided on a hybrid architecture that would combine the traditional Hadoop architecture and the local IT strategy.

A traditional Hadoop cluster consists of multiple, usually commodity grade, computational nodes each with local storage. The Hadoop system creates and manages a single file system (HDFS) distributed over all the local storage on the computational nodes. This makes it possible for Hadoop to assign the computational task to the node that is nearest the data. This is known as *data locality* and is one of the central principles of the Hadoop system.

In our hybrid we do not have data locality as none of our servers have any local storage. Instead each node is connected to the network attached storage (NAS) using NFS and on that storage create its part of the HDFS. Our NAS is the commercial Isilon Scale Out solution from the EMC company.

We considered using block storage but refrained from it as previous experiments showed that the NFS provides the necessary I/O performance. Still, block storage would give us higher performance and if necessary we will shift to that.

For this work we deployed four real servers connected to the scale out NAS solution. The specifications of the four servers are: Intel® Xeon® Processor X5670 (12M Cache, 2.93 GHz, 6.40 GT/s Intel® QPI). Each server has 2 CPUs each with 6 cores (24 cores per server if hyper threading is taken into account), 2×1 Gbit network interfaces, and 96GB of RAM.

This gives a Hadoop cluster with 96 hyper threaded cores, 384GB RAM, and 14TB HDFS storage.

The SCAPE project selected the Hadoop distribution from Cloudera as the target distribution. For the SB cluster we decided to use Cloudera Manager[34] in version 4.5 which deploys Hadoop version 2.0.0.

As the deployed Hadoop cluster is for experimentation only no focus has been put on server nor service redundancy e.g. we deploy no secondary name node. All four servers run both data nodes and task trackers and one of the servers in addition acts as name node and job tracker.

We have benchmarked this cluster using some of the standard Hadoop benchmark tests distributed with Cloudera. Some of our benchmark results are shown in table 1 for reference only.

Table 1: Benchmark results for the SB Hadoop cluster.

| Metric | Benchmark type | Value |
|---|---|---|
| Write throughput | TestDFSIO | 80MB/s |
| Read throughput | TestDFSIO | 278MB/s |
| Execution time | TeraSort | 2m 48s |

The TeraSort was run on 10GB of input data.

The network share where the data files from the Bit Repository can be read is on the same NAS that houses the HDFS. As such, reading data files should be comparable in speed to reading from HDFS.

# 5. THE CASE STUDY

The preservation action for the case study is feature extraction and profile validation of JP2 images from the newspaper collection described in section 5.1 below. It is based on the work flow illustrated in Figure 2.

1. Use the Stager to retrieve a number of File Objects from DOMS as METS documents to the Hadoop processing platform.
2. Start the Hadoop job, which, for each METS document will
   (a) run Jpylyzer on the referenced data file to extract features,
   (b) normalise a small set of these features to a generic characterisation format,
   (c) validate the normalised features against a collection specific control policy,
   (d) write the extracted features back into the METS document as technical metadata,
   (e) write the validation result back into the METS document as technical metadata.
3. Use the Loader to store the updated METS document in DOMS.

In order to perform this workflow, we need a few additional components, namely the SCAPE DOMS Connector that allows DOMS to be accessed through the SCAPE repository API and the feature extraction and validation job which is the core of this workflow. These will be described in the following sections.

## 5.1 Newspaper Digitisation Project

At SB, we are currently working on *Projekt Avisdigitalisering* (Newspaper Digitisation Project) [35]. In this project 32 million newspaper pages from the danish newspaper collection [36] will be digitised during the years 2013-2016. The pages are digitised from microfilm by Ninestars Information Technologies Ltd [37]. The goal is to provide free online access to as many pages as possible while showing the proper considerations for copyright holders.

The newspaper pages are digitised as lossless JP2 and estimated to require approximately half a Petabyte when the project is completed. The JP2 format was chosen as the best format for digital preservation in the prelude of the project [38].

The digitised pages will be stored in the Bit Repository and the metadata will be stored in the DOMS. The ingest is done as a minimal-effort-ingest, and the QA is then performed on data in the Bit Repository and DOMS. The QA involves a Jpylyzer characterisation of the image files and a policy driven validation of the image properties. It also involves an extensive policy driven validation of the metadata, as well as abnormality checks and a manual QA process.

## 5.2 SCAPE DOMS Connector

Implementing the SCAPE Repository API, described in section 3.3, could seem like a straightforward task for DOMS. DOMS, however, does not enforce a specific data model on its collections. Rather, one must specify a data model in the description of the collection. The challenge is thus allowing access to the various DOMS collections expressed in the uniform SCAPE Data Model.

A common building block of DOMS data models are File Objects, which correspond closely to SCAPE File Objects as described in section 3.2, in that they have exactly one reference to a data file and contain technical metadata about this file. Representations and Intellectual Entity Objects are not, however, a universal part of DOMS data models.

We have many collections in our repository. A requirement from our organisation is that these collections should not be altered in order to fit the SCAPE Data Model.

Two major collections in our repository that we want to access through the SCAPE DOMS Connector are the radio/tv collection and the newspaper collection described in section 5.1. The data models for these two collections are quite different.

The radio/tv collection can be viewed as two interlinked collections. First, there is the file collection, consisting of recorded broadcasts File Objects. Each File Object covers one or more radio/tv channels and a time span, irrespective of what programs were broadcast in that time span. Alongside that collection we have the program collection. It consists of Program Objects, with metadata about the program that was broadcast. Based on the start and stop time of the program we have then linked it to File Objects that cover the same time span.

The newspaper collection has a somewhat easier data model. The main object is the Newspaper Edition Object. Each Newspaper Edition Object has Page Objects. Each Page Object metadata has a File Object. The File Object has a URL to the file and technical metadata about the file. The Page has technical metadata about the scanning of the page, along with descriptive metadata about the page and OCR information. The Edition Object has descriptive metadata about the Edition as a whole.

We have chosen to use the File Object as the common link and to map each DOMS File Object to an entire SCAPE Intellectual Entity. This Intellectual Entity will have just one Representation, and this Representation will have just one File with no Bitstreams, making it more akin to figure 3, rather than figure 1. This mapping works for both the radio/tv collection and the newspaper collection.

All DOMS objects can have several named metadata records called data streams. The content models in a collection describe which data streams are required and the format of its records. In the collection specific content models for DOMS File Objects, we have specified which data streams should map to which metadata records in the SCAPE Data Model.

In this way we can maintain the different metadata records of each collection while allowing them to be mapped to and from the SCAPE Data Model.

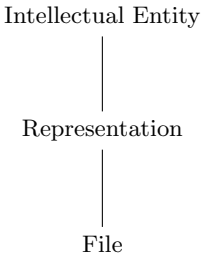


Figure 3: The SCAPE Model when used with DOMS.

Since only the DOMS File Objects are exposed in the SCAPE Data Model, it is clear that we will not ingest any new collections through this interface. Rather the SCAPE DOMS Connector will be used to add metadata to existing collections.

Some of our collections have multiple metadata records for each file. We thus need to be able to handle multiple technical metadata records in the SCAPE Data Model. We therefore enhanced the SCAPE Data Model and created a change request that is currently being processed be the SCAPE project.

## 5.3 Feature Extraction and Validation

The feature extraction and validation job builds upon the MapReduce programming model, using Jpylyzer and SCAPE Control Policies. The source code for the implementation is available from GitHub[1].

The input to the job is METS documents retrieved from the DOMS repository using the Stager on top of the SCAPE DOMS connector. The METS documents contain references to the JP2 image files in the Bit Repository.

The feature extraction and validation process is implemented as a single MapReduce job that extracts features by invoking Jpylyzer on the referenced file and validates these features. As the extracted features will be compared with corresponding metrics from the SCAPE Control Policy vocabulary, these features need to be mapped to that vocabulary. The SCAPE project is in the process of creating a generic framework for this type of mapping and we await that for a full solution to this challenge. When the extracted features are in the same vocabulary as the measures from the SCAPE Control Policy the MapReduce job can perform a comparison and validate the JP2 file accordingly.

The complete set of extracted features as well as the validation outcome, are stored in the METS documents, which is written back to the DOMS repository.

To prove the feasibility of the solution, we need not validate all the significant features of the images. The full list of format requirements for the images in the newspaper collection can be found in our wiki [39]. For this case study, we have chosen to focus on just one aspect, namely the colour profile and depth. The specification requires the colour profile to be greyscale and the colour depth to be 8 bit. We have identified which fields in the Jpylyzer XML output correspond to these two features. From this, we created a very specialised mapping of just the needed features from the Jpylyzer output to the SCAPE measures vocabulary, disregarding everything else.

[1] `http://github.com/statsbiblioteket/SCAPE-jp2-qa`

## 5.4 Experiment

The aforementioned Hadoop job has been executed on the SB Execution platform. We have run the hadoop job on sets of greyscale JP2 files from our repository. To verify that we could find both valid and invalid files, we added ten files with the RGB colorspace to each of the sets. The files were always correctly identified as not adhering to the preservation policy.

The performance results of running the experiment on the SB Hadoop cluster is shown in table 2 and in figure 4 we see that the execution time is close to linear and actually improving with the input size. For comparison we also executed the job as a standalone Hadoop job on a single commodity laptop computer which gave us the results shown in table 3.

From table 2 we see that on average, each node will spend about 0.15 seconds on each file. If we had to process the entire newspaper collection (32 million pages) on just a single node it would take more than 1200 hours. Hadoop provides the ability to reduce this total execution time by adding more nodes. Our experimental cluster consists of four nodes, and would spend about 300 hours on this job.

We have created our own implementations of the Stager and Loader clients for this case study, but as these are not the official SCAPE products, we do not want to show performance measurements. When the SCAPE project releases full versions of these we will evaluate the performance of the complete workflow, including the checkout and commit steps.

Table 2: Timing for JP2 image processing on the SB Hadoop cluster

| data size | #files | execution time | seconds per file per node |
|---|---|---|---|
| 167 GB | 17978 | 0h 14m 55s | 0.20 |
| 835 GB | 89890 | 0h 59m 23s | 0.16 |
| 1670 GB | 179780 | 1h 58m 08s | 0.16 |
| 4000 GB | 539340 | 5h 13m 35s | 0.14 |

Table 3: Execution time for the Hadoop job run on a commodity laptop

| Data size | #files | Execution time |
|---|---|---|
| 167 GB | 17978 | 1h 55m 11s |
| 835 GB | 89890 | 9h 14m 40s |

# 6. CONCLUSION

We showed that with a few adjustments the SCAPE Data Model can bridge the gap between our digital repository and the Hadoop platform for massive parallel preservation action execution.

As the DOMS data models are very heterogeneous we were unable to support the complete functionality of the SCAPE Data Model. This limitation is primarily on ingesting new

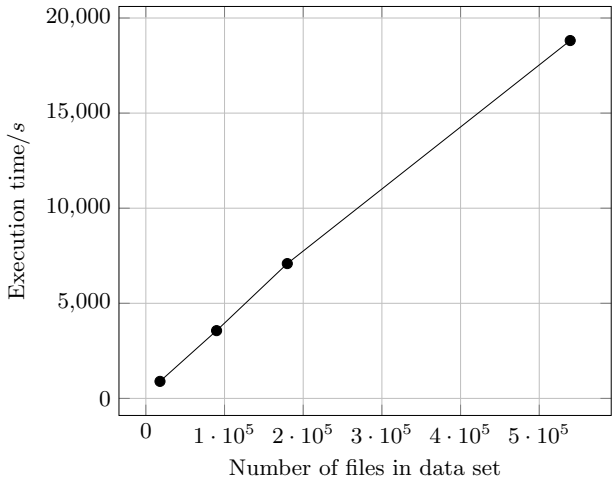


Figure 4: Execution times of the experiment on the SB Hadoop cluster as described in section 4.3

data described by the complete SCAPE Data Model. If we only use the proposed system for performing preservation actions on already ingested data, this restriction is of minimal consequence.

We implemented a stager and a loader using the SCAPE Data Connector API and have shown a feasible path for adding massive parallel preservation action execution to our existing digital repository and as a case study we validated half a million JP2 files against an institutional preservation policy.

The preservation action in our case study was policy driven validation of JP2 files. We chose a single measure, namely the color profile, from our institutional preservation policy for scanned newspaper pages and validated that each JP2 file was grayscale with 8 bit colour depth. We used the Jpylyzer tool and showed that on the implemented platform the preservation action preforms linear in time as the data amount grows to 4TB.

The SCAPE project is in itself a good example of how to scale. To take up the challenge of massively parallel digital preservation we need to work together across institutions and borders.

## 6.1 Further Work

We will continue working on the SCAPE Data Model to ensure it will meet our requirements in getting metadata out of and into our repository and improve our support of the SCAPE Data Model.

The SCAPE Stager and Loader components are not yet available from the SCAPE Project. We have developed our own version of these[2], which should either be given back to SCAPE or abandoned in favour of the official components when they are released.

Further experiments should be done with more resource demanding preservation actions, both with regard to CPU load, I/O load, and complexity. A first step would be to employ the Nanite tool [40] instead of Jpylyzer.

A crucial new development is the creation of a framework for mapping from specific tool output to the SCAPE measures and control policy vocabulary. This will make it easy to take a new tool and use it for validation against an institutional preservation policy.

[2] `https://github.com/statsbiblioteket/scape-stager-loader`

# 7. ACKNOWLEDGEMENTS

We thank the following for their invaluable help in discussing and proofreading this paper: Bjarne Andersen, Jette Junge, Karen Williams, and Kåre Fiedler Christiansen. We thank Tom Gravgaard Christensen and Jens Henrik Leonard Jensen for creating the Hadoop cluster. Last but certainly not least we thank all of our colleagues in the SCAPE project for all the great discussions on large scale challenges and solutions.

# 8. REFERENCES

[1] (2014, Mar.) Doms. live wiki page. [Online]. Available: http://sbforge.org/display/DOMS/Home

[2] (2014, Mar.) The bit repository project. live wiki page. [Online]. Available: http://sbforge.org/display/BITMAG/The+Bit+Repository+project

[3] (2014, Mar.) Scalable preservation environments. The SCAPE project. [Online]. Available: http://www.scape-project.eu

[4] (2014, Mar.) Welcome to apache™ hadoop®! Apache Software Foundation. [Online]. Available: http://hadoop.apache.org

[5] (2014, Mar.) Radio/tv-samlingen. In Danish. State and University Library. [Online]. Available: http://www.statsbiblioteket.dk/nationalbibliotek/adgang-til-samlingerne/tv-og-radio/radio-tv

[6] (2014, Mar.) Bl 19th century digitized newspapers. Live wiki page. The SCAPE project. [Online]. Available: http://wiki.opf-labs.org/display/SP/BL+19th+Century+Digitized+Newspapers

[7] (2014) The netarkivet web site. [Online]. Available: http://netarkivet.dk/in-english

[8] (2014, Mar.) Fedora repository project. Fedora Commons. [Online]. Available: http://fedoracommons.org

[9] (2014, Mar.) Islandora web site. Islandora Foundation. [Online]. Available: http://islandora.ca

[10] (2014, Mar.) Hydra web site. The Hydra Project. [Online]. Available: http://projecthydra.org

[11] (2014, Mar.) escidoc project web site. [Online]. Available: http://www.escidoc.org/

[12] (2014, Mar.) Dspace is a turnkey institutional repository application. Duraspace. [Online]. Available: http://www.dspace.org

[13] (2014, Mar.) Eprints - digital repository software. [Online]. Available: http://www.eprints.org

[14] (2014, Mar.) darceo web site. [Online]. Available: http://dingo.psnc.pl/darceo/

[15] (2014, Mar.) The lily web site. [Online]. Available: http://www.lilyproject.org/lily/index.html

[16] (2014, Mar.) Rosetta web site. ExLibris. [Online]. Available: http://www.exlibrisgroup.com/category/RosettaOverview

[17] (2014, Mar.) Roda web site. [Online]. Available: http://www.roda-community.org/

[18] *Audit and Certification of Trustworthy Digital Repositories*, ser. Magenta Book, No. 1, Consulative

Committee for Space Data Systems (CCSDS) Recommandation for Space Data Systems Standard 652.0-M-1, September 2011.

[19] (2014, Mar.) Published preservation policies. Live wiki page. The SCAPE Project. [Online]. Available: http://wiki.opf-labs.org/display/SP/Published+Preservation+Policies

[20] M. Sheldon. (2013, Aug.) Analysis of current digital preservation policies: Archives, libraries and museums. [Online]. Available: http://blogs.loc.gov/digitalpreservation/2013/08/analysis-of-current-digital-preservation-policies-archives-libraries-and-museums

[21] (2014, Mar.) Bl 19th century digitized newspapers. Live wiki page. The SCAPE project. [Online]. Available: http://http://wiki.opf-labs.org/display/SP/LSDRT2+EX1+BL+Newspapers+on+the+BL+Platform

[22] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[23] (2014, Mar.) Scape platform. live wiki page. The SCAPE project. [Online]. Available: http://wiki.opf-labs.org/display/SP/SCAPE+Platform

[24] M. Hahn, F. Asseg, N. Sherwinter, , and R. Castro. Scape data model. The SCAPE Project. [Online]. Available: http://github.com/openplanets/scape-apis/blob/master/Digital_Object_Model_V1.0.pdf

[25] M. Hahn, F. Asseg, , N. Sherwinter, and R. Castro. (2014, Jan.) Recommendations for preservation-aware digital object model. To be published on http://scape-project.eu.

[26] (2014, Mar.) The premis data dictionary for preservation metadata. Library of Congress. [Online]. Available: http://www.loc.gov/standards/premis

[27] (2014, Mar.) Mets, metadata encoding and transmission standard. Library of Congress. [Online]. Available: http://www.loc.gov/standards/mets

[28] M. Hahn and F. Asseg. Scape connector api. The SCAPE Project. [Online]. Available: http://github.com/openplanets/scape-apis/blob/master/Data_Connector-API_V1.1.pdf

[29] B. Sierman, C. Jones, S. Bechhofer, and G. Elstrøm, "Preservation policy levels in scape," in *iPRES 2013 – Proceedings of the 10th International Conference on Preservation of Digital Objects*, 2013.

[30] S. Bechhofer, B. Sierman, C. Jones, G. Elstrøm, H. Kulovits, and C. Becker. (2014) Final version of policy specification model. The SCAPE Project. [Online]. Available: http://http://www.scape-project.eu/deliverable/d13-2-catalogue-of-preservation-policy-elements-draft

[31] J. van der Knijff. Jpylyzer, jp2 validator and extractor. National Library of the Netherlands. [Online]. Available: http://openplanets.github.io/jpylyzer

[32] Iso/iec 15444-1:2004 information technology – jpeg 2000 image coding system: Core coding system. International Organization for Standardization. [Online]. Available: http://www.iso.org/iso/catalogue_detail.htm?csnumber=37674

[33] (2014, Mar.) Enhanced content models for fedora commons 3.x. [Online]. Available: http://wiki.duraspace.org/display/FCREPO/Enhanced+Content+Models

[34] (2014, Mar.) Cloudera manager web site. Cloudera. [Online]. Available: http://http://www.cloudera.com/content/cloudera/en/products-and-services/cloudera-enterprise/cloudera-manager.html

[35] (2014) Om projekt avisdigitalisering. State and University Library. [Online]. Available: http://en.statsbiblioteket.dk/national-library-division/newspaper-digitisation/newspaper-digitization

[36] (2014, Mar.) Statens avissamling. In Danish. State and University Library. [Online]. Available: http://www.statsbiblioteket.dk/nationalbibliotek/adgang-til-samlingerne/aviser/StatensAvissamling

[37] (2014, Mar.) Ninestars web site. [Online]. Available: http://ninestar.co.in

[38] (2014, Mar.) Choosing file format for digital preservation. State and University Library. [Online]. Available: http://blog.avisdigitalisering.dk/format/#Choosing

[39] A. Askov-Blekinge. (2013) Jpeg2000 specifications for the newspaper collection. State and University Library. [Online]. Available: https://sbforge.org/display/NEWSPAPER/Appendix+2B+-+JPEG2000+specifications

[40] (2014, Jul.) Nanite web site. [Online]. Available: https://github.com/openplanets/nanite