An OAIS-oriented System for Fast Package Creation, Search, and Access

Sven Schlarb, Rainer Schmidt, Roman Karl, Mihai Bartha, Jan Rörden
AIT Austrian Institute of Technology
Donau-City-Straße 1
1220 Vienna, Austria
{first}.{last}@ait.ac.at

Janet Delve
University of Brighton
CRD, Grand Parade
Brighton, BN2 0JY, UK
BN2 4AT
J.Delve@brighton.ac.uk

Kuldar Aas
National Archives of Estonia
J. Liivi 4
Tartu, 50409, Estonia
kuldar.aas@ra.ee

ABSTRACT

This paper describes the scalable e-archiving repository system developed in the context of the E-ARK project. The system is built using a stack of widely used technologies that are known from areas such as search engine development, information retrieval and data-intensive computing, enabling efficient storage and processing of large volumes of data. The E-ARK Integrated Platform Reference Implementation Prototype takes advantage of these technologies and implements an OAIS-oriented repository system for creating, archiving, and accessing data as information packages. The system consists of software components including an efficient file handling infrastructure, a configurable and scalable ingest system, a powerful full-text-based search server, and a distributed repository providing file-level random access. This paper gives an overview of the architecture and technical components that have been used to build the prototype. Furthermore, the paper provides experimental results and gives directions for future work.

CCS Concepts

•Information systems → Data management systems;
•Applied computing → Document searching;

Keywords

OAIS; archiving; repository; scalability; distributed systems; Hadoop

1. INTRODUCTION

In recent years, considerable research and development efforts dealt with managing the growing amount of digital data that is being produced in science, information technology, and many other areas of today's society [9]. The constant increase in the number of digital publications, governmental records, or digitized materials is challenging for the development of procedures and information systems for libraries and archives [10]. An effort to cope with preservation workflows that need to be executed on large volumes of digital materials has been made by the SCAPE project [12], which has developed a platform that enables users to execute such processes using computer clusters and data-intensive computing techniques [19]. The E-ARK Integrated Platform Reference Implementation Prototype[1] continues this work by setting up a scalable repository system for archival institutions.

The integrated prototype has been developed in the context of the E-ARK project, an ongoing 3-year multinational research project co-funded by the European Commission's ICT Policy Support Program (PSP) within the Competitiveness and Innovation Framework Program (CIP). The purpose of the integrated prototype is to demonstrate how open source solutions for distributed storage and processing can be combined to build a scalable repository for memory organizations. The aim is to show that this approach is, in general, suitable to address the need for enhancing existing archiving systems in providing access to very large, continuously growing, and heterogeneous digital object collections in archival institutions.

In its first project year, E-ARK has conducted a GAP analysis among archival institutions identifying user requirements for access services[2]. The study investigated the current landscape of archival solutions regarding the available access components and identified gaps and requirements from the perspective of national archives, 3rd party users, as well as content providers. The study identified a major gap in the identification process where users browse and search collections to identify material of potential interest. It stated that a lack of comprehensive metadata available and indexed compromises the performance and efficiency of the finding aids, which directly impacts the user experience and the user's access to the archival holdings in their entirety.

To fill this gap, E-ARK makes use of s scalable repository system and search infrastructure for archived content. The goal is not necessarily to replace existing systems but to augment these components (like archival catalogues) with a "content repository" that can be searched based on a full text index. The content repository concentrates on fine

iPres '16 October 3–6, 2016, Bern, Switzerland
© 2016 ACM. ISBN 978-1-4503-2138-9…$15.00
DOI: 10.1145/1235

[1]in the following shortly called "integrated prototype".
[2]http://www.eark-project.com/resources/project-deliverables/3-d51-e-ark-gap-report

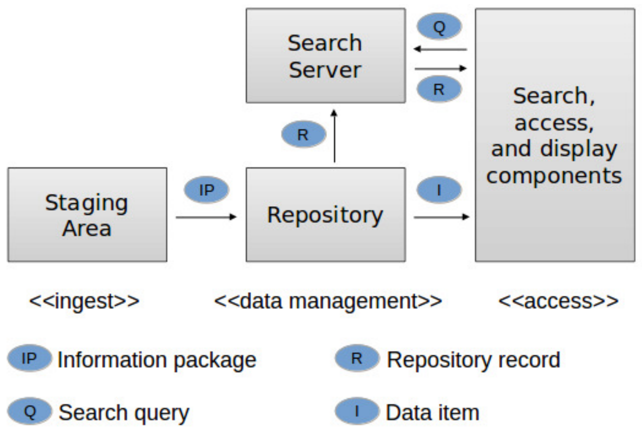


Figure 1: System Components and their interactions used by the integrated prototype for implementing the Faceted Query Interface and API.

grained search within information packages and random access at the file-level rather than providing search based on selected metadata elements and package-based access. The integrated prototype developed in this context employs scalable (cluster) technology as scalability issues must be taken into account when operating a detailed content-based search facility, providing an infrastructure for creating, ingesting, searching, and accessing E-ARK information packages. Scalability is accomplished by making use of technologies like the Apache Hadoop framework[3], NGDATA's Lily repository[4], and the Apache SolR search server[5].

The workflow implemented by the integrated prototype for data ingest, storage, and access is based on the ISO Reference Model for an Open Archival Information System (OAIS) [3]. This means that data is received in form of Submission Information Packages (SIPs) which are transformed into Archival Information Packages (AIPs) and transferred to the archive. Upon client request the selected content of the AIPs can be retrieved from the archive and repacked as Dissemination Information Packages (DIPs) for delivery. The repository supports facet search based on full-text and extracted metadata (e.g. MIME-type, size, name of the files contained within the information packages). This is accomplished by executing information extraction and transformation processes upon transfer of the SIP to the archive (SIP to AIP conversion/ingest).

2. BACKEND ARCHITECTURE

2.1 Overview

Figure 1 provides an overview of the major system components that are employed by the backend of the integrated prototype. The query interface and API provided by the search server must be backed by software components and generated data products in order to provide the desired functionality. Here, we give an overview and describe their interactions.

[3]http://hadoop.apache.org/
[4]https://github.com/NGDATA/lilyproject
[5]http://lucene.apache.org/solr/

2.2 Staging Area

The staging area is a file-system based storage location provided in combination with the data management component of the integrated prototype. The staging area is accessible to other components based on an API allowing these components to deposit information packages for ingestion into the content repository (as shown in Figure 1). While in principle any file system could be employed as staging area, the integrated prototype makes use of the Hadoop File System (HDFS) for performance, scalability and reliability reasons. The staging area is in the first place used to access the information packages during the repository ingest workflow but can also be employed to serve other purposes like (package-based) access.

2.3 Repository

The integrated prototype makes use of NGDATA's Lily project which is employed as a content repository. The information packages residing on the staging area are ingested into the repository where they are stored in the form of structured repository records, as described in section 3. The repository interacts with the search server which reads and indexes the repository records as well as with client components which access data items on a file or record level.

2.4 Search Server

The generation and/or update of the index provided by the search server can be triggered by the repository component in case records are added, deleted, or modified. The index provides the necessary data structure to evaluate search queries and to return results which point to records stored in the repository. The index and search functionality is provided by the search server through an HTTP interface. The integrated prototype makes use of Apache Solr as the search server which can be well integrated with Lily and its underlying database HBase. The query interface is provided by a defined REST API through Apache Solr which is customized based on the individual structure of the repository records. For supporting multiple and heterogeneous collections, it is possible to generate different indexes for different datasets maintained by the repository.

2.5 Search, Access, and Display Components

These components interact with the search server and the repository as clients. Specific archival user interface and access components (e.g. required for DIP creation) have been implemented in the context of the E-ARK Web project, as described in section 5.2. The protocol for interacting with the query interface is however independent of the employed client component and ultimately allows for the integration with an external user interface. Client components typically provide a graphical representation of the query language and facets provided by the search server. When a query is submitted to the search server, it is evaluated against the index. The search server subsequently returns a ranked list of record references (and optionally content fragments) to the client. Besides interfaces required for searching, the repository also provides an access service providing clients with random access to data on a file-level, based on references, which can retrieved by an HTTP request, issued for example through the client application.

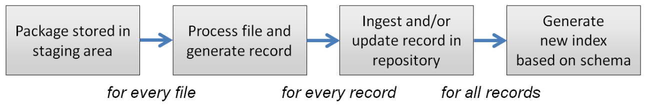

Package stored in staging area
Process file and generate record
Ingest and/or update record in repository
Generate new index based on schema
for every file
for every record
for all records

Figure 2: Conceptual workflow for ingestion and indexing of information packages to the content repository provided by the integrated prototype.

3. CONCEPTUAL WORKFLOW

Figure 2 shows the conceptual workflow for ingesting data items residing on the staging area (for example using the Hadoop File system) into the content repository. Practically, this means that after the repository has been populated and/or updated a full text index is generated and/or updated respectively.

The integrated prototype implements the ingest workflow for ingesting information packages into the content repository on a file-based level which is in contrast to ingesting on a package level. The ingest workflow is implemented in a way that every item (or file) contained within an information package is considered a record. In the repository, packages are represented as a set of records sharing a common identifier.

3.1 Record extraction and ingest

Once the ingest process is started, the workflow iterates over all files contained within the individual information packages. Each file extracted from the information package is processed separately. The exact implementation of the processing step is highly depending on the data set and institutional requirements. Examples that have been implemented as part of the integrated prototype include the extraction of text portions, structure, and context information from web, office, or XML documents, file size calculation, MIME-type identification, and checksums.

The data extracted from an individual file is subsequently stored into a data structure, called a record , which can be ingested into the repository. The individual structure of a record can be customized depending on data and institutional needs. A record for a newspaper article, for example, could contain fields like author, title, body, publisher, and publishing date. Fields of a record are “typed” which means they can be restricted to certain data types like for example numbers, string, or date. A record identifier that encodes the identifier of the package as well as the location of the original file within the package is generated automatically. Once a record is created, it is ingested into the content repository. As records organize information in a structured way, they can be interpreted by the repository and consequently stored in a (structured) database.

3.2 Full-text index generation

The E-ARK integrated prototype aims at providing a facet query interface based on full-text indexing in addition to the rather limited search mechanisms provided through database indexing. The full-text search functionality is provided through a search server (like Apache Solr), which relies on a previously generated full-text index (using Apache Lucene). The integrated prototype makes use of a configuration file (called a schema) that provides a detailed specification of the indexing process. This controls for example which parts of a record should be indexed, available fields, and the information that should be stored with the index (e.g. only document references and/or also content portions).

After new content has been ingested and/or updated the repository index should be generated or updated at periodic intervals. The integrated prototype provides specific commands for triggering the creation of the index from the records available within the repository. Depending on the volume of content, indexing as well as ingestion can become very resource and time consuming processes. Both processes have therefore been implemented as parallel applications that can take advantage of a computer cluster to scale out for large data sets. Within the E-ARK integrated prototype, indexing and ingestion have been deployed on a cluster at AIT, providing a total of 48 CPU-cores. The generated index is made available by the search server as a query interface enabling a client to formulate and execute queries against the index, compose complex queries based on facets, and rank them based on different characteristics. It is however important to note that although a defined query API is exposed by the integrated prototype, the API is highly configurable and customizable with respect to the parameters it accepts and the nature of results it returns.

The workflow shown in Figure 2 was implemented based on the software components described in section 2 (and shown in Figure 1). It has been configured for different test data and deployed in a single-node environment as well as in a cluster environment available at AIT.

	Master	*Slave*
HDFS	NameNode	DataNode
MapReduce	JobTracker	TaskTracker
HBase	HBase Master	Region Server

Table 1: Daemons running on the cluster.

4. SCALABLE PROCESSING AND SEARCH INFRASTRUCTURE

Although systems for parallel and distributed computing have been studied since the early 1980’s and parallel database systems were established already in the mid-1990’s [1], a significant change in the last decade occurred with the advent of the MapReduce data processing paradigm [5] and the subsequent rise of open source technology for distributed storage and parallel data processing provided by Apache Hadoop. In the following, we describe the integrated prototype backend which is based on Apache Hadoop and related components that emerged in the Hadoop ecosystem during the last decade.

4.1 Hadoop

The backend system of the integrated prototype is built on top of the Hadoop framework and can be deployed on a computer cluster allowing the repository infrastructure to scale-out horizontally. This enables system administrators to increase the available system resources (i.e. for storage and processing) by adding new computer nodes. Using Hadoop, the number of nodes in a cluster is virtually unlimited and clusters may range from single node installations to clusters comprising thousands of computers.

Usually one would, however, build a cluster consisting of a master node and at least two slave nodess to get a performance advantage from the distributed environment. Each slave machine runs all services, which means that it runs a DataNode, a TaskTracker and a Region Server. For production clusters, it is recommended to deploy the NameNode on its own physical machine and furthermore use a Secondary-NameNode as a backup service. Although Lily is deployed on multiple nodes, it does follow the concept of master and slave nodes. There is only one type of Lily node which is intended to run co-located with Region Servers on the cluster.

4.2 Lily

Lily provides a repository that is build on top of HBase, a NoSQL database that is running on top of Hadoop. Lily defines some data types where most of them are based on existing Java data types. Lily records are defined using these data types as compared to using plain HBase tables, which makes them better suited for indexing due to a richer data model. The Lily Indexer is the component which sends the data to the Solr server and keeps the index synchronized with the Lily repository. Solr neither reads data from HDFS nor writes data to HDFS. The index is stored on the local file system and optionally distributed over multiple cluster nodes if index sharding or replication is used. Solr can be run as a standalone Web-based search server which uses the Apache Lucene search library for full-text indexing and search. The integrated prototype utilizes the Lily Java API as part of a Hadoop MapReduce job in order to ingest large volumes of files in parallel.

4.3 Solr

There are several options to run Solr. The first option is to run Solr only on one machine. In this case the index is not split and only one shard is used. The second option is to use multiple shards and configure Lily to distribute the input over all shards. As Solr 4 introduced SolrCloud, this became the third option, and it is also the preferred option for a production system. SolrCloud does not only take care of the sharding, it also provides a mechanism for replication. Using Lily in combination with SolrCloud requires some additional configuration work being done, as Lily was developed against Solr 4.0, where SolrCloud was not yet entirely mature. For an example, it is required to create an empty directory in ZooKeeper manually where SolrCloud can store its information.

4.4 ZooKeeper

HBase, but also Lily and SolrCloud, depend on a running ZooKeeper cluster. ZooKeeper is a framework that supports distributed applications in maintaining configuration information, naming, providing distributed synchronization, and providing group services. ZooKeeper stores small amounts of information, typically configuration data, which can be accessed by all nodes. For experimental clusters that do not need to provide high fault tolerance, it is sufficient to run one ZooKeeper node, which is also called *Quorum Peer*. A higher fault tolerance can be achieved by running three, five or more Quorum Peers. If more than half of the nodes keep running without failures, ZooKeeper stays reliable.

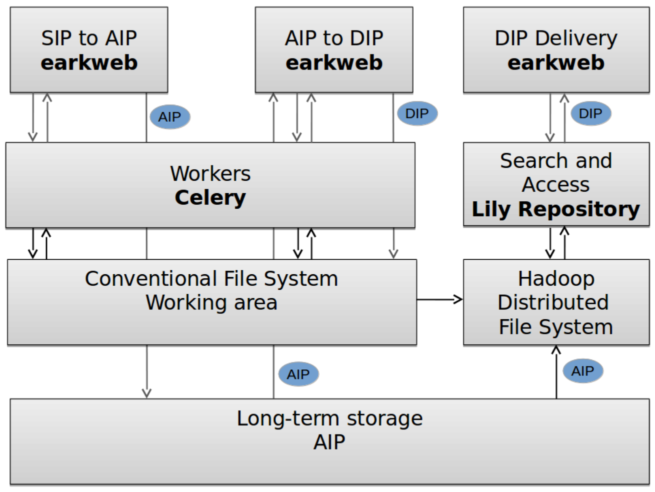


Figure 3: The architecture consists of user interface components that support information package ingest and access processes. The frontend components are backed by a package creation infrastructure handling file, task, and workflow processing. The frontend system is integrated with the Hadoop backend infrastructure for content extraction, storage, and search.

5. FRONTEND ARCHITECTURE

5.1 Overview

In general, the backend system of the integrated prototype takes information packages as input and provides functionalities like information extraction, search, and random access for the contained data items. In the previous chapters, we have outlined a set of custom components and services which have been specifically developed to realize the integrated prototype. The E-ARK Web Project described in section 4 provides a lightweight front-end implementation for this backend system. The responsibility of the frontend system is the provisioning of user interfaces and corresponding services for creating information packages like AIPs and DIPs.

The architecture consists of user interface components that support information package ingest and access processes. The frontend components are backed by a Package Creation Infrastructure handling file, task, and workflow processing. The frontend system is integrated with the Hadoop backend infrastructure for content extraction, storage, and search. The implementations provided by the integrated prototype are lightweight applications which are limited in their functionality and focused on distinct E-ARK principles. The architecture of the integrated prototype is in general designed to support a loose coupling strategy so that existing systems can be combined with and/or be augmented with the integrated prototype or particular components of the integrated prototype platform.

5.2 The E-ARK Web Project

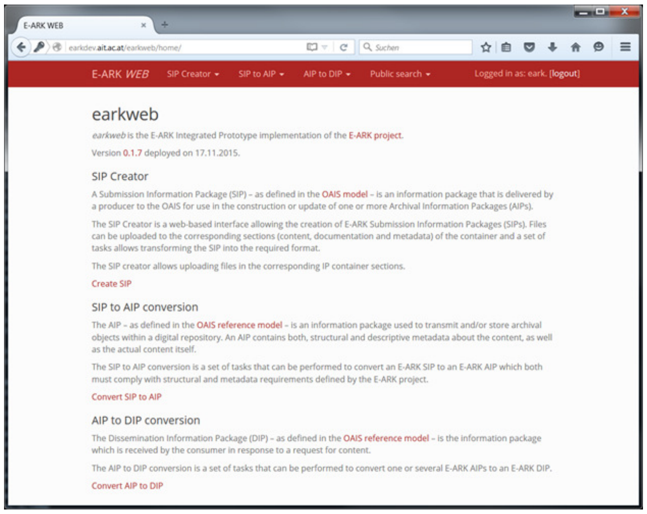
The project *E-ARK Web* [6] is a web application together with a task execution system which allows synchronous and asynchronous processing of information packages by means of processing units which are called "tasks". The purpose of E-ARK Web is, on the one hand, to provide a user interface for the integrated prototype in order to showcase archival information package transformation workflows which are being developed in the E-ARK project in an integrated way. On the other hand, the goal is to provide an architecture which allows reliable, asynchronous, and parallel creation and transformation of E-ARK information packages (E-ARK SIP, AIP, and DIP) integrated with E-ARK backend services for scalable and distributed search and access.

The components of the E-ARK Web project coordinate package transformations between the package formats SIP, AIP, and DIP, and uses Celery [7], a distributed task queue, as its main backend, shown in figure 3. Tasks are designed to perform atomic operations on information packages and any dependency to a database is intentionally avoided to increase processing efficiency. The outcome and status of a task's process is persisted as part of the package. The E-ARK Web project also provides a web interface that allows one to orchestrate and monitor tasks by being loosely coupled with the backend. The backend can also be controlled via remote command execution without using the web frontend. The outcomes of operations performed by a task are stored immediately and the PREMIS format [2] is used to record digital provenance information. It is possible to introduce additional steps, for example, to perform a roll-back operation to get back to a previous processing state in case an error occurs.

5.3 The E-ARK Web User Interface

The user interface of the integrated prototype is a Python[8] /Django[9]-based web application which allows for managing the creation and transformation of E-ARK information packages (E-ARK IPs). It supports the complete archival package transformation pipeline, beginning with the creation of the Submission Information Package (SIP), over the conversion to an Archival Information Package (AIP), to the creation of the Dissemination Information Package (DIP) which is used to disseminate digital objects to the requesting user. The E-ARK Web website is divided into four main areas: First, there is the "SIP creator" area which allows initiating a new SIP creation process and offers a set of transformation tasks to build E-ARK compliant SIPs. Second, there is the "SIP to AIP" area that allows for the execution of tasks for converting an E-ARK compliant SIP to an AIP. Third, there is the "AIP to DIP" area which allows initiating a DIP creation process based on previously selected AIPs used for building the DIP with the help of a set of corresponding conversion tasks. And, finally, there is the "Public search" area offering full-text facet search based on the textual content available in the AIPs which have been uploaded to the HDFS staging area, ingested into Lily, and full-text indexed using SolR, as described in section 3. A screenshot of this user interface is shown in Figure 4.

[6]https://github.com/eark-project/earkweb
[7]http://www.celeryproject.org
[8]https://www.python.org
[9]https://www.djangoproject.com



earkweb

earkweb is the E-ARK Integrated Prototype implementation of the E-ARK project.

Version 0.1.7 deployed on 17.11.2015.

SIP Creator

A Submission Information Package (SIP) – as defined in the OAIS model – is an information package that is delivered by a producer to the OAIS for use in the construction or update of one or more Archival Information Packages (AIPs).

The SIP Creator is a web-based interface allowing the creation of E-ARK Submission Information Packages (SIPs). Files can be uploaded to the corresponding sections (content, documentation and metadata) of the container and a set of tasks allows transforming the SIP into the required format.

The SIP creator allows uploading files in the corresponding IP container sections.

Create SIP

SIP to AIP conversion

The AIP – as defined in the OAIS reference model – is an information package used to transmit and/or store archival objects within a digital repository. An AIP contains both, structural and descriptive metadata about the content, as well as the actual content itself.

The SIP to AIP conversion is a set of tasks that can be performed to convert an E-ARK SIP to an E-ARK AIP which both must comply with structural and metadata requirements defined by the E-ARK project.

Convert SIP to AIP

AIP to DIP conversion

The Dissemination Information Package (DIP) – as defined in the OAIS reference model – is the information package which is received by the consumer in response to a request for content.

The AIP to DIP conversion is a set of tasks that can be performed to convert one or several E-ARK AIPs to an E-ARK DIP.

Convert AIP to DIP

Figure 4: The earkweb user interface showing the four main areas SIP creator, SIP to AIP, AIP to DIP and Public search.

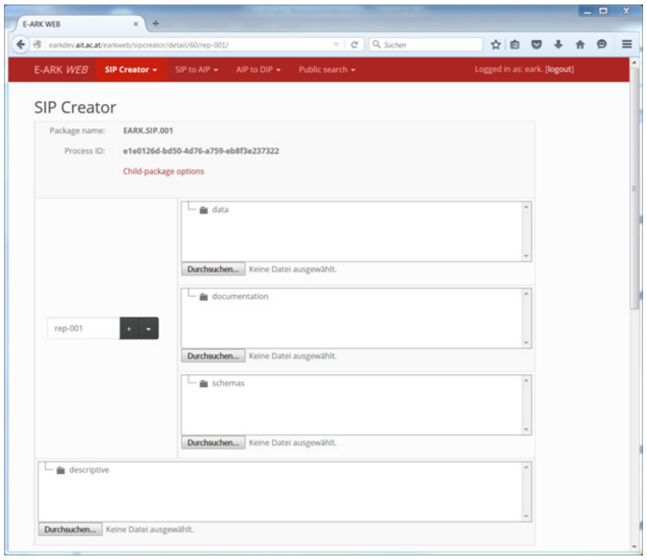
The common denominator of the "SIP creator", "SIP to AIP", and "AIP to DIP" areas is that they all offer information package transformation tasks. The transformation of information packages is implemented in the same way across all of the three information package transformation areas. The "SIP creator" and the "AIP to DIP" areas additionally provide some basic setup forms in order to collect information needed to initiate a new process. As shown in Figure 5, the "SIP creator" provides a form which allows for uploading individual files into the corresponding areas of the information package.

The interface for executing tasks is basically the same across all package transformation areas. The difference lies in the tasks they provide. Figure 6 shows the task execution interface of the "SIP to AIP" conversion. The pull-down select field shows tasks that are available in this area. Here, the available tasks are related to information packages which are converted from the initially submitted SIP to the AIP, which is finally transmitted to the long-term storage and/or uploaded into the distributed storage area for full-text indexing and access.

Figure 7 shows a search interface used in the "AIP to DIP" dialog that allows one to discoverer data in AIPs, select individual items, and generate DIPs.

5.4 Asynchronous and parallel package processing

As mentioned in section 5.3, the transformation of information packages is implemented in the same way across all of the three information package transformation areas. In this section, we describe the task execution infrastructure used by the E-ARK Web project to enable the reliable and controlled execution of information package transformation tasks. Apart from the Python/Django-based user interface, E-ARK Web uses a backend for asynchronous and parallel task execution based on the Celery task execution system,



SIP Creator

Package name:	EARK.SIP.001
Process ID:	e1e0126d-bd50-4d76-a759-eb8f3e237322

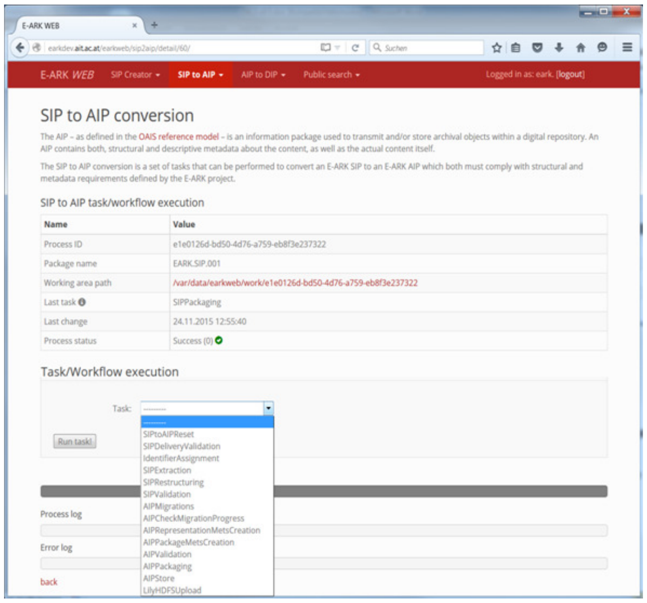
data — Keine Datei ausgewählt.

documentation — Keine Datei ausgewählt.

schemas — Keine Datei ausgewählt.

descriptive — Keine Datei ausgewählt.

Figure 5: User Interface of the SIP creator providing a form to select individual files and the corresponding location within the information package.



SIP to AIP conversion

The AIP – as defined in the OAIS reference model – is an information package used to transmit and/or store archival objects within a digital repository. An AIP contains both, structural and descriptive metadata about the content, as well as the actual content itself.

The SIP to AIP conversion is a set of tasks that can be performed to convert an E-ARK SIP to an E-ARK AIP which both must comply with structural and metadata requirements defined by the E-ARK project.

SIP to AIP task/workflow execution

Name	Value
Process ID	e1e0126d-bd50-4d76-a759-eb8f3e237322
Package name	EARK.SIP.001
Working area path	/var/data/earkweb/work/e1e0126d-bd50-4d76-a759-eb8f3e237322
Last task	SIPPackaging
Last change	24.11.2015 12:55:40
Process status	Success (0) ✓

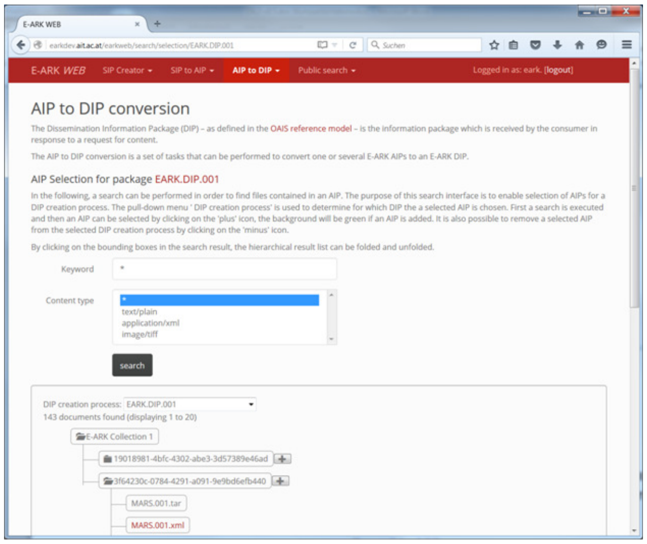
Task/Workflow execution

Task: ---------

Process log

Error log

Figure 6: User interface for selecting and starting an information package transformation. This screenshot shows the SIP to AIP conversion area.


E-ARK WEB
earkdev.ait.ac.at/earkweb/search/selection/EARK.DIP.001
Suchen
E-ARK WEB
SIP Creator
SIP to AIP
AIP to DIP
Public search
Logged in as: eark. [logout]
search
E-ARK Collection 1
19018981-4bfc-4302-abe3-3d57389e46ad
3f64230c-0784-4291-a091-9e9bd6efb440
MARS.001.tar
MARS.001.xml

AIP to DIP conversion

The Dissemination Information Package (DIP) – as defined in the OAIS reference model – is the information package which is received by the consumer in response to a request for content.

The AIP to DIP conversion is a set of tasks that can be performed to convert one or several E-ARK AIPs to an E-ARK DIP.

AIP Selection for package EARK.DIP.001

In the following, a search can be performed in order to find files contained in an AIP. The purpose of this search interface is to enable selection of AIPs for a DIP creation process. The pull-down menu ' DIP creation process' is used to determine for which DIP the a selected AIP is chosen. First a search is executed and then an AIP can be selected by clicking on the 'plus' icon, the background will be green if an AIP is added. It is also possible to remove a selected AIP from the selected DIP creation process by clicking on the 'minus' icon.

By clicking on the bounding boxes in the search result, the hierarchical result list can be folded and unfolded.

Keyword	*
Content type	[✓ *] text/plain application/xml image/tiff

DIP creation process: EARK.DIP.001

143 documents found (displaying 1 to 20)

Figure 7: Search interface in the AIP to DIP dialogue allowing a user to discover and select relevant data items from AIPs available in the Content Repository.

the MySQL[10] database, and the RabbitMQ[11] message broker software.

Whenever a task is initiated using the E-ARK Web task execution interface, the RabbitMQ message broker receives a message which is subsequently consumed by the Celery task execution engine. Tasks can be assigned to workers which are configured in the Celery backend. The workers share the same storage area and the result of the package transformation is stored in the information package's working directory based on files.

As the actual status of the transformation process is persisted during the task execution it is not required to interrupt the processing chain for every executed task in order to update status information in the database. Based on the results stored in the working directory, the status of an information package transformation can be updated with a single operation when the transformation process has finished. This strategy increases the processing efficiency, which is critical when large volumes of data are processed, and helps avoiding bottlenecks caused by a large number of parallel database connections. Another advantage of this approach is that by design it is possible to reconstruct the databases, tracking the status of the processed information package, based on the information contained in the working directories. Particular importance was given to the principle of avoiding to instantly record digital object related processing information in the database as this may entail the risk of significantly increasing the processing time for very large information packages.

The decision to use either synchronous or asynchronous task execution for a specific task depends on the type of task and also the kind of data the information package contains. A task which itself initiates an unknown number of sub-tasks, can lead to a long task runtime, possibly beyond

[10]https://www.mysql.com
[11]https://www.rabbitmq.com

the defined timeout limit. An example would be a set of file format migration sub-tasks which are triggered for specific file types, e.g. each PDF file contained in an information package is converted to PDF/A. These cases can be implemented using a master task that starts an unknown number of sub-tasks and records the amount of migrations to be performed. This task is followed by a verification task which can be executed manually or automatically to report the current status of task executions. This way, it is possible to control that subsequent executions are not started before all sub-tasks were executed successfully, and that all the (possibly long-running) processes are decoupled from each other. The upload of an AIP into the Hadoop infrastructure has been implemented as a synchronous task. The live progress of the upload process is shown directly in the user interface. However, if for cases where AIPs tend to be very large – where "large" is to be seen in the context of available bandwidth and read/write throughput – it is easily possible to change this task execution into an asynchronous task

5.5 Task and Workflow Definition

With respect to software design, a major goal was to foster flexibility, modularity, and extensibility of the task execution base class. Tasks are implemented in one single Python script and only contain the code that is necessary for the concrete task implementation. The intention is to keep the actual task implementation slim and offload extensive functionality into an earkcore Python module[12] which can be made available to the Celery workers.

The E-ARK Web project defines a workflow model on top of the task execution layer. The "state" of an information package, as described earlier, is defined by storing the "last executed task" together with the success/failure of the execution. Tasks provide interface definitions (like for example "allowed inputs") which provide the basis for workflow composition. Using this information together with the current execution status, the workflow engine can control if a task is allowed to be performed on a specific information package.

New tasks can be easily added to the system by supplying a new task class implementation based on a Python script. The new task is available in the system as soon as the Celery workers are re-initialized. The configuration of the task is handled directly within the task implementation based on code annotations. Information to verify workflow composition is immediately available through the task description and does not require any additional configuration files. As the descriptive information is used to initialize the task configuration information in the database, it can be also dynamically adapted in the database, if required.

6. EXPERIMENTAL EVALUATION

6.1 Hardware environment

The Lily/Hadoop deployment on the development cluster at AIT is shown in figure 8. The cluster comprises seven physical machines which are structured into a master and six physical slave nodes. Each node on the cluster provides 6 CPU cores (12 threads using Intel HT), 16GB RAM and 16TB SATA (hotplug) of storage. Each cluster node is equipped with two network interfaces allowing us to attach

[12]https://github.com/eark-project/earkweb /tree/master/earkcore

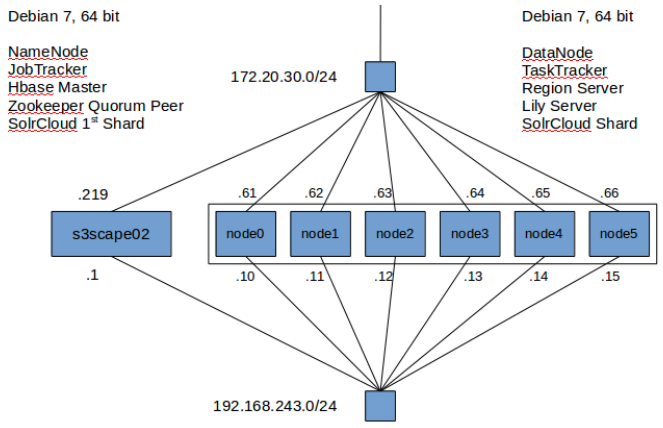


Figure 8: Hardware cluster at AIT used to host a Lily repository on top of Hadoop, HDFS and HBase.

a node to two network infrastructures. The cluster is connected to the internal network allowing us to directly access each node from desktop/working environments. The second private network is used for managing the cluster. For example, new cluster nodes can be automatically booted and configured using the PXE pre-boot execution environment together with a private Fully Automated Install (FAI) server[13].

6.2 Data set

The govdocs1 corpus [8] is a set of about 1 million files that are freely available for research. This corpus provides a test data set for performing experiments using different types of typical office data files from a variety of sources. The documents were originally obtained randomly from web servers in the .gov domain. Due to the volume of collected files and the variety of data types available in this corpus, we have chosen to perform a document discovery over the entire corpus as a simple use case for evaluating for the E-ARK integrated prototype.

Here, it is important to note that the integrated prototype is designed for the ingestion of information packages as described by the OAIS model. E-ARK is developing a general model along with as set of specifications and tools for handling SIP, AIP, and DIP packages, which are being included with the integrated prototype's package creation infrastructure. AIPs are typically created as structured tar-files containing data and metadata as described by the E-ARK AIP format[14]. The repository provided by the integrated prototype is designed to maintain the structure of the ingested information packages (by encoding file locations within the record identifier) — allowing users to browse and search single packages if desired — but in general provides search and access across information packages on a per-file basis. For the experimental evaluation we have ingested the govdocs1 corpus in the form of 1000 tar files, each containing 1000 documents, which results in 1000 packages available in the integrated prototype's repository, and 1 million files that are full-text indexed, and that can be individually identified by an URL and accessed via the REST API.

[13]http://fai-project.org

[14]http://www.eark-project.com/resources/project-deliverables/53-d43earkaipspec-1

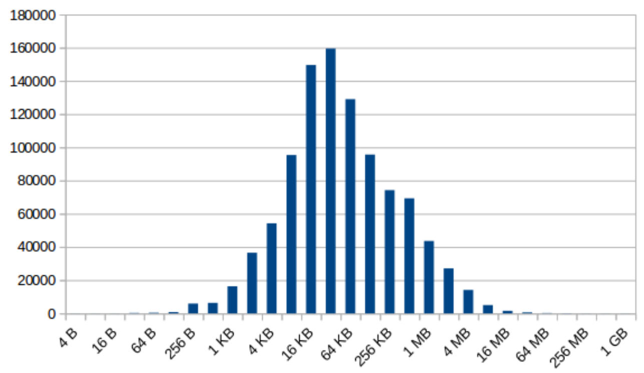


Figure 9: Govdocs1 file size distribution

TestDFSIO	*write*	*read*
Number of files	10	10
Total MBytes processed	10000.0	10000.0
Throughput MB/sec	17.80604586481	49.9852543499
Average IO rate MB/sec	17.93034553527	52.1930541992
Test exec time sec	72.661	36.593

Table 2: I/O performance benchmarking

Hadoop Job	*File Ingest Mapper*
Number of map tasks	1000
Map input records	984951
Map output records	354
Job finished in	1hrs, 47mins, 51sec

Table 3: The integrated prototype automatically triggers a MapReduce job when ingesting data into the repository. The table shows the results reported by the Hadoop MapReduce execution environment after the govdocs 1 corpus has been ingested as a set of 1000 tar-files.

Query parameter	*Value*
facet	on
q	*:*
facet.field	contentTypeFixed
rows	0

Table 4: Parameters of a faceted query that orders the search results by the number of by MIME-types.

Once all packages are ingested, documents can be found using the search server API. Queries might include a full text search string, filtering based on metadata (like MIME-type), or restrict the search results on certain packages. Using facets in a search query allows one to easily derive general statistics about a search result. Figure 9 illustrates the result of a faceted search query which groups all files of the ingested govdocs1 corpus based on file-sizes. Most of the files fall in the range between 1KB and 16MB and only a few small files with size values starting from 7 bytes and 4 text files over 1.5 gigabytes exist. An overview of the MIME types available in the corpus is described by [17, p. 15]. We will show as part of this evaluation how to retrieve this kind of information from the system once the collection has been successfully ingested.

6.3 Cluster I/O benchmarking

To provide indicative benchmarks, we executed the Hadoop cluster I/O performance benchmarking test “TestDFSIO” as described by [15] which is a read and write benchmarking test for the Hadoop Distributed File System (HDFS). TestDFSIO is designed in such a way that it uses 1 map task per file. This is similar to the file ingest component of the integrated prototype where each package (available as a TAR file) is processed by one task. The default test method of TestDFSIO is to generate 10 output files of 1GB size for a total of 10GB in the write test which are subsequently read by the “read” test. The results of this test are as presented in table 2.

6.4 Evaluation results

The purpose of this evaluation is to give an approximate insight on the performance of the E-ARK integrated prototype. Due to the complexity of the system set-up and the numerous configuration options, the presented results should only provide an indication of the achieved cluster performance rather than provide strict benchmarking results.

We defined a threshold for the file ingest workflow (executed as a map task) to process a maximum file size of 50 Megabytes. The Govdocs1 corpus contains 354 files exceeding this limit. These files sum up to a total size of about 42 Gigabytes and were ingested separately. The pre-configured file limitation is an implementation detail which has been set for practical reasons. In case it is required to automatically ingest files of large sizes, this can be handled as well. While Lily stores small files in HBase for efficient random access, large files are stored directly in HDFS. There is no file size limitation regarding the ingest or storage of files in the repository. The basic test results of the Hadoop job performing the ingest workflow are shown in table 3.

The number of 1000 map tasks corresponds to the 1000 TAR packages of the Govdocs1 corpus which were defined as the input of the Hadoop job. The 984951 input records are the individual files which were found in the TAR packages. The map task performs the ingest of files into Lily and outputs only those files which had been skipped due to their file size, as described earlier. The set of 1000 processed tar-files sums up to a total of 467GB and the total wall time for the ingest process amounts to 1 hour and 47minutes.

The files contained in the ingested tar-files are searchable using the Solr search interface. Part of the job execution was to run text extraction and MIME-Type detection using Apache Tika and to store this information in the index, therefore it is now possible to run a single faceted Solr query to get basic MIME-Type statistics with the parameters specified in table 4, where the field “contentTypeFixed” is the field of type “string” defined in the schema of the collection which holds the MIME-type of the corresponding file item. This allows us, for example, to get an overview about the ten most frequent MIME types in the collection as presented in figure 10.

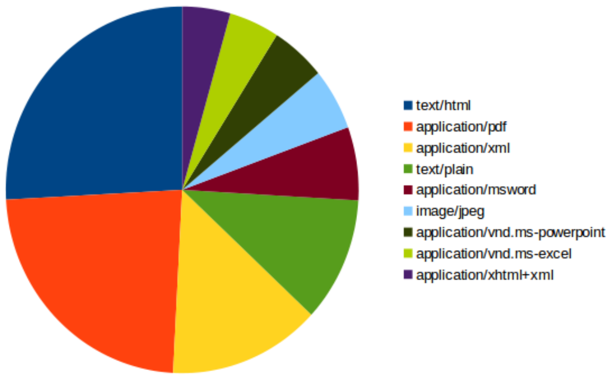


Figure 10: The ten most frequent MIME types (resulting from a Solr facet query)

7. ADVANCED DATA ACCESS SCENARIOS

As part of the ingest workflow, the integrated prototype adds, besides full-text information, various (often content and use-case specific) metadata elements to the the repository record in order to enhance the capabilities of the derived search index, as briefly demonstrated in the previous section. Ongoing experimental work is dealing with utilizing additional data mining strategies to further enhance the integrated prototype's search functionality. Two currently evaluated text mining strategies are Classification and Named Entity Recognition, as explained below. The goal is to add useful information to the full-text index, such as discovered text categories. Besides selecting appropriate algorithms and designing useful features, it is a challenge to run such data mining algorithms on very large data volumes. Initial experiments have been performed using ToMar [20], a MapReduce application for efficiently running 3rd party tools on a Hadoop-based cluster, developed in the context of the SCAPE project.

7.1 Text classification

The classification of text allows one to make assumptions on the contents of files, based on a previously trained model. We are planning to use this technique to extend the search interface provided by the E-ARK Web user interface through adding a field for selecting automatically recognized text categories. This way it is possible to search for documents which relate to a specific topic combined with the existing query and filtering options provided by the search server. The number of search-able topics depends on the previously trained classifier process, and therefore include an assumption on which topics could be of interest for the user. As a toolkit for implementing text classification, we have utilized the *scikit-learn* [16] Python framework.

7.2 Named Entity Recognition

An additional goal was to identify locations, persons, or other terms of interest as so called Named Entities. In initial tests the the Stanford Named Entity Recognizer[7] has been utilized to extract entities from text documents. Entities that were classified as locations were, in an additional step, geo-referenced using the Nominatim database [4]. As a result, an XML file containing a list of found locations together with their corresponding coordinates was generated for each analyzed document. The intention behind this work is to incorporate new ways of making archived content accessible to the user. Initial experiments dealt with visualizing the geographical focus of identified topics over time using the graphical map annotation tool Peripleo [21].

8. RELATED WORK

Warcbase [13] uses HBase as the core technology to provide a scalable and responsive infrastructure for web archiving. The environment makes use of the random access capabilities of HBase to build an open-source platform for managing raw content as well as metadata and extracted knowledge. Additionally, Warcbase provides exploration, discovery, and interactive visualization tools that allow users to explore archived content.

The Internet Memory Foundation has built a distributed infrastructure for Web archiving, data management, and preservation using Apache Hadoop as one of the core technologies [14]. Their focus is on scalability issues in terms of crawling, indexing, preserving and accessing content.

RODA is an open source digital repository which delivers functionality for all the main units of the OAIS reference model [6]. RODA provides distributed processing and the execution of digital preservation actions (e.g. migration) on a Hadoop cluster.

The European project SCAPE (Scalable Preservation Environments) addressed the preservation of very large data sets found in digital repositories, scientific facility services, and web archives as one of the main use cases [18]. SCAPE has build on top of a Hadoop-based infrastructure for defining and carrying out preservation workflows. Additionally the project investigated the integration of an Hadoop-based infrastructure with the Fedora Commons repository systems [11].

9. CONCLUSIONS

In this paper we presented a prototype infrastructure for the scalable archiving of information packages developed in the E-ARK project. The system is implemented using a set of open source technologies including for example Apache Hadoop, the Lily Project, and Apache SolR. As we have outlined, there are a number of related projects, mostly in the Web archiving domain, which are using a similar technology stack for scalable e-archiving. The system presented in this paper is however targeting the archival community and specifically designed to support OAIS-based concepts. The Integrated Platform Reference Implementation Prototype has been developed to handle the creation, ingestion, and access of E-ARK information packages using an environment that scales from a single host to a cluster deployment. The system can be deployed as a stand-alone environment but also next to existing archiving systems in order to enhance available services, like for example finding aids (using the full text index) and order management (using the content repository).

Here, we have provided a brief overview of the system architecture and the employed technologies. We have also described the ingest workflow in more detail and explained how the individual components are employed and how they are related to each other. As an evaluation of the approach, we have ingested and indexed the entire Govdocs1 corpus consisting of nearly 1 million documents with a total size of about 467 Gigabytes in less then 2 hours, making the text content discoverable in and across information packages based on full-text as well as metadata-based queries using a powerful search server. The used repository provides instant access at the granularity of single files which can be viewed and/or packaged for dissemination using the provided E-ARK Web access components. The paper reports also on future directions to further improve the search capabilities of the system by employing data mining algorithms.

10. ACKNOWLEDGMENTS

Work presented in this paper is primarily supported by European Community's Seventh Framework Program through the E-ARK project under grant agreement number 620998.

11. REFERENCES

[1] V. Borkar, M. J. Carey, and C. Li. Inside "big data management": Ogres, onions, or parfaits? In *Proceedings of the 15th International Conference on Extending Database Technology*, EDBT '12, pages 3–14, New York, NY, USA, 2012. ACM.

[2] P. Caplan and R. S. Guenther. Practical preservation: the premis experience. *Library Trends*, 54(1):111–124, 2006.

[3] CCSDS. *Reference Model for an Open Archival Information System (OAIS)*. CCSDS - Consultative Committee for Space Data Systems, January 2012. Version 2 of the OAIS which was published in June 2012 by CCSDS as "magenta book" (ISO 14721:2012).

[4] K. Clemens. Geocoding with openstreetmap data. *GEOProcessing 2015*, page 10, 2015.

[5] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. In *OSDI'04: Proceedings of the 6th conference on operating systems design and implementation*. USENIX Association, 2004.

[6] L. Faria, M. Ferreira, R. Castro, F. Barbedo, C. Henriques, L. Corujo, and J. C. Ramalho. Roda - a service-oriented repository to preserve authentic digital objects. In *Proceedings of the 4th International Conference on Open Repositories*. Georgia Institute of Technology, 2009.

[7] J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *In ACL*, pages 363–370, 2005.

[8] S. Garfinkel, P. Farrell, V. Roussev, and G. Dinolt. Bringing science to digital forensics with standardized forensic corpora. *digital investigation*, 6:S2–S11, 2009.

[9] A. J. Hey, S. Tansley, K. M. Tolle, et al. *The fourth paradigm: data-intensive scientific discovery*, volume 1. Microsoft research Redmond, WA, 2009.

[10] A. J. Hey and A. E. Trefethen. The data deluge: An e-science perspective. 2003.

[11] B. A. Jurik, A. A. Blekinge, R. B. Ferneke-Nielsen, and P. Møldrup-Dalum. Bridging the gap between real world repositories and scalable preservation environments. *International Journal on Digital Libraries*, 16(3):267–282, 2015.

[12] R. King, R. Schmidt, C. Becker, and S. Schlarb. Scape: Big data meets digital preservation. *ERCIM News*, 89:30–31, 2012.

[13] J. Lin, M. Gholami, and J. Rao. Infrastructure for supporting exploration and discovery in web archives. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14 Companion, pages 851–856, New York, NY, USA, 2014. ACM.

[14] L. Medjkoune, S. Barton, F. Carpentier, J. Masanès, , and R. Pop. Building scalable web archives. In *Archiving Conference, Archiving 2014 Final Program and Proceedings*, number 1, pages 138–143, 2014.

[15] M. Noll. Benchmarking and stress testing an hadoop cluster with terasort, testdfsio & co. http://www.michael-noll.com/blog/2011/04/09/benchmarking-and-stress-testing-an-hadoop-cluster-with-terasort-testdfsio-nnbench-mrbench, 4 2011.

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[17] M. Radtisch, P. May, A. A. Blekinge, and P. Moldrup-Dalum. D9.1 characterisation technology, release 1 and release report. http://www.scape-project.eu/wp-content/uploads/2012/03/SCAPE_D9.1_SB_v1.0.pdf, 04 2012.

[18] S. Schlarb, P. Cliff, P. May, W. Palmer, M. Hahn, R. Huber-Moerk, A. Schindler, R. Schmidt, and J. van der Knijff. Quality assured image file format migration in large digital object repositories. In J. Borbinha, M. Nelson, , and S. Knight, editors, *iPRES 2013: 10th International Conference on Preservation of Digital Objects, 2-6 September 2013*, pages 9–16, Lisbon, Portugal, September 2013. Lisbon Technical University (IST).

[19] R. Schmidt. An architectural overview of the scape preservation platform. In *9th International Conference on Preservation of Digital Objects*, pages 85–55. Citeseer, 2012.

[20] R. Schmidt, M. Rella, and S. Schlarb. Constructing scalable data-flows on hadoop with legacy components. In *11th IEEE International Conference on e-Science, e-Science 2015, Munich, Germany, August 31 - September 4, 2015*, page 283, 2015.

[21] R. Simon, E. Barker, L. Isaksen, and P. de Soto Cañamares. Linking early geospatial documents, one place at a time: Annotation of geographic documents with recogito. *e-Perimetron*, 10(2):49–59, 2015.