Noname manuscript No.
(will be inserted by the editor)

Digital Preservation in Grids and Clouds: A Middleware Approach.

Peter Wittek · Sándor Darányi

Received: date / Accepted: date

Abstract Digital preservation is the persistent archiving of digital assets for future access and reuse, irrespective of the underlying platform and software solutions. Existing preservation systems have a strong focus on grids, but the advent of cloud technologies offers an attractive option. We describe a middleware system that enables a flexible choice between a grid and a cloud for ad-hoc computations that arise during the execution of a preservation workflow and also for archiving digital objects. The choice between different infrastructures remains open during the lifecycle of the archive, ensuring a smooth switch between different solutions to accommodate the changing requirements of the organization that needs its digital assets preserved. We also offer insights on the costs, running times, and organizational issues of cloud computing, proving that the cloud alternative is particularly attractive for smaller organizations without access to a grid or with limited IT infrastructure.

Keywords Digital Preservation · Grid · Cloud

1 Introduction

Preservation of digital assets requires more constant and ongoing attention than preservation of other media, because such collections are volatile and can be catastrophically lost much more easily and quickly than physical assets due to technical and human failures [39]. Digital preservation (DP) combines policies, strategies and actions to ensure that digital objects remain authentic and accessible to users and systems over a long period of time, regardless the challenges of component and management failures, natural disasters or attacks [5]. This includes the preservation of materials resulting from digital reformatting, but particularly of information that is born-digital and has no analogue counterpart.

Distributed DP methodologies state that any responsible preservation system must distribute copies of digital assets to geographically dispersed locations. A single organization with preservation needs is unlikely to have the capability to operate geographically dispersed and securely maintained servers. Hence collaboration between institutions is essential, and this collaboration requires both organizational and technical investments [39]. Long-term inter-institutional agreements must be put in place, and these typically translate to distributed grids shared by the participating institutions. As a consequence, much of the research on technologies that support DP focuses on grids.

DP, however, is not just about the secure backup of the bitstream of digital assets, it is also about future access and reuse. While DP has a vast literature exposing the wide array of associated issues, here we would like to focus on three aspects only: migration and transformation, scalability, and reusability. The first aspect refers to the problem of keeping the content of legacy file formats accessible. This problem is most prominent with proprietary file formats for which documentation is not available. Once the vendor stops support for the associated software products, these files face digital obsolescence. A common solution to the problem is migration in which older formats are transformed to a

This work was funded by Sustaining Heritage Access through Multivalent ArchiviNg (SHAMAN), a large-scale integrating project by the European Union (Grant Agreement No. ICT-216736)

Peter Wittek · Sándor Darányi
Swedish School of Library and Information Science
University of Borås
Borås, Sweden

more persistent format. Secondly, dynamic collections and environments for DP require technical scalability to face technology evolution. Existing static collections, for instance, a digitised historical archive, where no new items will be added, will have a fixed data size. Although it will not be a must to add new components to increase the storage capacity, it may be necessary to replace components or transform the objects in the collection. These requirements ask for scalability [4]. Achieving this with DP may require specific investments in an infrastructure for storing, maintaining, and managing data. Such costs can be prohibitive for organizations whose core business is not data conservation and that do not have a considerable budget for investments in information technology. The third important aspect of DP that we would like focus on is to enable the reuse of digital content. Reuse of digital content covers its subsequent verification and exploitation with a novel purpose, potentially by new consumers. Since consumers are unable to refer back to the creators, reuse of preserved digital objects depends on proper descriptions provided through the archive [15]. In this sense, reuse of digital content asks for metadata on both the content and how it was transformed to its most recent form. This is where document process preservation helps, which provides an architecture-independent description of the intent behind a document process [24]. These three aspects of DP are not unrelated. The processing pipeline is computationally expensive, and to our knowledge this issue has not been addressed directly in the DP literature.

A fundamental and often overlooked problem is that organizations outside institutional or inter-institutional grids also require the preservation of their digital assets. The infrastructure offered by cloud providers can be a solution for them both for distributed preservation and for ad-hoc computations. Larger organizations may also study the cloud alternative to reduce cost or outsource archival-related activities.

In this paper, we build on existing solutions to derive a middleware approach that enables a smooth switch between a grid and a cloud infrastructure addressing both computational and storage needs. Both ad-hoc computations and archival can be performed either in a grid or a cloud. As the collection expands or the archiving organization changes, our solution offers a smooth transition between the grid and the cloud if the need arises. Relying on experiments in the cloud, and taking a few organizational issues in consideration, we conclude that the cloud alternative is particularly attractive to smaller institutions. The suitability of cloud storage as a means to achieve geographically distributed fault tolerant backup is well known, and therefore our focus is on the computational aspect.

This paper is organized as follows. Section 2 briefly overviews some concepts of digital preservation as these will be crucial for the rest of the paper. Section 3 touches upon related work with particular focus on existing and proven solutions for grid-based digital preservation. Taking these solutions as the starting point, our proposed flexible middleware for digital preservation in grids and clouds is described in detail in section 4. Section 5 further discusses the implementation and reveals some experimental results. After the main discussion, we highlight some non-technological issues that may influence the decision of whether an organization should choose a grid or cloud solution for preservation purposes (Section 6). Finally, Section 7 concludes the paper.

2 Requirements in digital preservation: the Open Archival Information System and beyond

The most popular standard in the area of DP is ISO 14721:2003 Space data and information transfer systems – Open archival information system – Reference model, widely known as OAIS [23]. It is a functional framework which presents the main components and basic data flows within a DP system. OAIS is intended to identify the necessary features of an archival information system rather than recommend any particular type of implementation. It also draws attention to the important role of preservation for repositories, asking that it should be considered alongside other functions and activities [1]. Institutional repositories are a particular form of digital archives that are implemented for use within an institutional setting. Institutional repositories often pay less attention to preservation [25]. To share essential functions and requirements of preservation between two or more institutions, OAIS establishes a common framework of terms and concepts, and comprises several models which prescribe a minimal set of responsibilities required for the preservation of digital objects [3]:

- The Information Model identifies three types of information packages, namely Archival Information Package, Submission Information Package, and Dissemination Information Package which address the encapsulation of information objects within OAIS. Information packages include content information composed from the data object and Representation Information and Preservation Description Information and additional packaging information.
- The Functional Model distinguishes between seven separate functional entities and their related inter-

faces. The main functions of an OAIS are modelled as seven functional entities.
– The Environment Model defines the generic roles interacting with the OAIS system, identifying three roles: Producers, Consumers and Management.

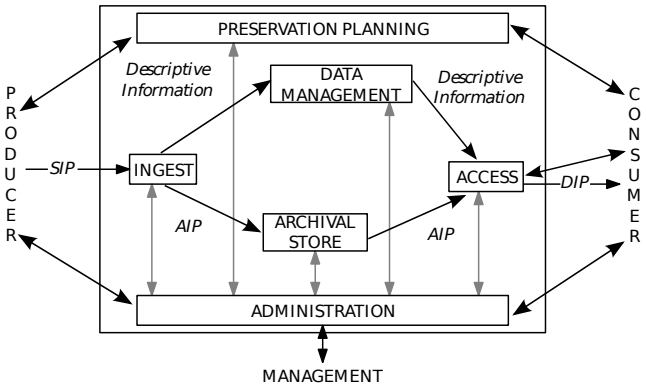

Fig. 1 OAIS functional model

The functional model is illustrated in Figure 1.

The view of OAIS is exclusively archival-centric, and therefore it is quite limited in scope when it comes to the complete lifecycle of a digital object. The European Commission cofunded the integrated project Sustaining Heritage Access through Multivalent ArchiviNg (SHAMAN) with the aim to investigate the long-term preservation of large volumes of digital objects in a distributed environment, by developing a preservation framework that is verifiable, open and extensible [22]. To understand the broader context of digital objects that need to be preserved, SHAMAN defines features not present in the OAIS [38], such as layered information packages where each layer is addressed by a particular preservation activity, activities that precede the ingest phase, activities that succeed the access (i.e., post-access) phase, and further refinement of the information package to ensure the information necessary to guarantee long-term preservation is included. The latter is particularly important because of the additional information required by the description of pre-ingest workflows and to facilitate post-access (see also Section 3.2).

The applied software architecture is driven by the requirements from OAIS and the additional points defined by SHAMAN and reflects the requirements towards a Service-oriented Architecture (SoA). This architecture consists of three layers (see Figure 2). The first one is the presentation layer [32]. Presentation services are provided through a web interface with Multivalent Media Engines for rendering the different data types. Multivalent is an extensible digital document

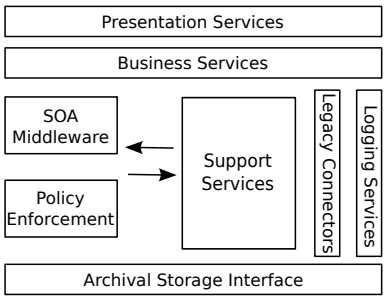

Fig. 2 Overview of the SHAMAN core infrastructure

framework to support any format, which will provide future access to data formats [33]. The middle layer is the Business layer which serves as a mediator between the presentation layer and the infrastructure. It holds the services and SoA components like the service orchestration and service registry. It also contains the policies that describe the agreement between the Producer and Archive Management on how the data are to be stored in the long-term archive such that they can be usable in the future by a defined group of Consumers. The lowest layer is the infrastructure layer at which the components of storage infrastructure shall be specified. Communication interfaces mediate the information flow between those different layers. Figure 2 illustrates the main components of the SHAMAN core infrastructure (see also the next section for implementation technologies of the framework and Section 4 for the proposed middleware that supports the Business Services layer).

3 Related technologies in digital preservation

3.1 Archiving to a grid

Data grids provide several functions required by DP systems, particularly when massive amounts of data must be preserved [5], offering a distributed infrastructure and services that support applications dealing with massive data collections stored in heterogeneous distributed resources. Focusing on the technical aspects first, grids are built using middleware making fundamental aspects such as file management, user management and networking protocols completely transparent. Other threats in a grid environment include organizational issues, such as management failures. These can be considered at a higher organizational level through preservation policies that include duplication of resources and integrity checks among others.

The Integrated Rule-Oriented Data System (iRODS, [34]) is a state-of-the-art open source distributed software system for addressing key data management tasks

taking a comprehensive approach to full data life cycle management. It is a middleware that manages a highly controlled collection of distributed digital objects, while enforcing user-defined management policies across the multiple storage locations. The SHAMAN framework uses iRODS as an implementation technology.

At the iRODS core, a rule engine interprets the rules to decide how the system is to respond to various requests and conditions. This ability to execute rules conditionally, and to define multiple rules implementing alternative means of achieving the same goal, provides a degree of flexibility that hold great promise for implementing automated digital curation and preservation applications [21, 20].

The iRODS server software is installed on each storage system where data will be stored. The remote location of the storage system is normally defined by an IP network addresss. The iRODS server translates operations into the protocol required by the remote storage system. In addition, a rule engine is also installed at each storage location. The rule engine controls operations performed at that site. The iRODS data grid effectively implements a simple distributed operating system.

The iRODS solution controls low-complexity workflows that can be most efficiently executed at each storage system. Examples of low-complexity workflows include the extraction of a data subset from a large file, or the parsing of metadata from a file header. The iRODS system interacts with remote computers to execute high-compexity tasks. This capability is currently enabled through support for encapsulation of remote web services as iRODS micro-services. Within the iRODS workflow, calls can be issued to remote processing systems which manipulate data that can be moved across the network.

3.2 Preservation workflows

Preserving the intent behind the activities and projects that led to the production of digital data can be as meaningful and important in the long term as preserving the data. Indeed, an important goal of DP is to enable reuse of digital content by securing the long term understanding of the intent behind its preservation. This is an integral part of the extensions of OAIS provided by SHAMAN.

Production and reuse of digital content from the archive do not coincide, as reuse may have a novel purpose or may operate in a totally different environment than was available at production time. Preservation must bridge gaps in time, space, semantics, knowledge, objectives and other dimensions. This is why the description of preserved digital objects must include means to understand the context in which the data was initially produced and used. This context is not only defined by the digital objects themselves, but also by the processes in which they were created, ingested, accessed and re-used.

The Xeproc[1] domain-specific language, developed within SHAMAN, addresses precisely the need to capture the intent behind document processes, so that they can be preserved and reused in future unknown infrastructures [24]. To that end, Xeproc preserves not only production processes, but also instrumented specifications of a document processing project. The SHAMAN framework uses Xeproc as an implementation technology.

Xeproc technology can be used to build a wide range of applications based on document processing, including transformation, extraction, indexing and navigation. It can be easily integrated with more global business processes and customized to match specific requirements and infrastructures. In the spirit of SoA, Xeproc embeds references to services and documents and provides loose coupling not only to services but also to data resources, with respect to both their location and format.

These capture the intent behind the workflow irrespective of the implementation at a given point in time (see Figure 3). Such abstract representations are preserved, so that the Xeproc models, utilizing XML, can be seen as independent specifications to be instantiated and deployed over time and as technology evolves.

XML is ideally suited to represent the logical structure of documents (e.g. titles, sections, chapters, paragraphs, reading flow) independently of their visual rendering. By explicitly encoding a document's structure and meaning, XML opens up the possibilities for document lifecycle management, including content reuse and repurposing, quality assurance and security. The sheer volume and heterogeneity of document collections (numerous authoring systems and proprietary formats such as PDF, PS, Word, and TIFF), as well as the nature and characteristics of the information to be encoded (data-oriented documents such as purchase orders, complex and implicit structures such as maintenance manuals), makes document conversion to XML a complex and delicate task.

XML processing often involves transformations from one XML format to another. Extensible Stylesheet Language Transformations (XSLT) is a declarative, XML-based language used for describing these transformations [42]. For instance, XSLT is often used to convert

[1] Available on Eclipse 3.5.1 under the Eclipse Public License at `http://marketplace.eclipse.org/content/xeproc`

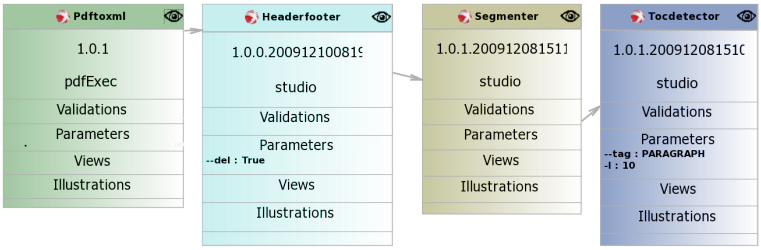

Fig. 3 Document process designer showing a process to extract table of contents

XML data into HTML or XHTML documents for display as a web page. In the context of DP, content may be stored in a richly annotated XML format, and when it is presented to the user, an XSLT transformation can be used to render the page in a legible format.

Within the context of SHAMAN, Xeproc has been specifically used to model XML processing pipelines and XML validation checkpoints. Validation checkpoints may be defined in any of several schema languages such as Document Type Definition or Relax NG. Those XML processing pipelines focus on identifying structural metadata describing the document organization. The pipelines eventually produce metadata in Metadata Encoding and Transmission Standard (METS, [10]) addressing various aspects of the document collections, such as extraction of logical organization, extraction of the page numbering (for book navigation), and extraction of illustrations and associated captions. Eventually, the packaged pipelines can be deployed towards a production platform.

3.3 Retrieval

Designing and developing digital library access services focus on approaches to indexing and searching in an increasing range of genres and materials. An important aspect of this research is concerned with providing effective and scalable information retrieval services for digital libraries and archives as these diverse collections continue to grow.

The Cheshire project has a research focus on large-scale digital collections with a focus on supporting distributed digital libraries in a grid environment [26]. At the same time the project has been prototyping systems for very long term digital preservation, and examining how grid-scale information retrieval systems can interoperate with petabytes of diverse data stored over many years. The Cheshire system implements a set of objects with precisely defined roles that permit digital collection operations to be distributed over many nodes on a network, vastly increasing the throughput of data for compute and storage intensive processes with little overhead beyond single processor solutions. Once the distributed infrastructure has been defined, one or more 'master' workflows divide the processing requirements among a number of 'slave' processes. The implementation uses the Parallel Virtual Machine as a very fast communication layer over a gigabit switched network. The Parallel Virtual Machine is a software tool for parallel networking of computers, allowing a network of machines to be used as a single distributed parallel processor [40]. When a slave process completed its workflow, it returns the results to the master to be merged. Object interactions within the architecture just like results from any workflow are well-defined, based on the last processing object. This is particularly important for trees or graphs of distributed workflows. If the index files at each node store all of the terms for the records at the node, then interacting with the subset of the database is very fast, but the intermediate result sets will need to be merged before a global query can be answered. If each node maintains a portion of the terms for all of the records, then it can answer the query authoritatively, requiring central authority to manage the division of the index terms.

As part of the ongoing efforts in SHAMAN, Xeproc pipelines and components were packaged into standalone Python programs that can be deployed on any node of a grid. In the project, iRODS was used for storage with Cheshire used for full-text indexing. Cheshire proved to be a fast engine for digital libraries in a distributed grid environment [26,27], and integrates well with iRODS [44].

3.4 Additional support services

As part of the support services in the SHAMAN core infrastructure, several natural language processing and data mining techniques were considered to produce useful tools for accessing and reusing preserved digital assets.

Data mining processes typically include supervised learning such as classification (predicting if a given document is a member of a particular class), and unsu-

pervised learning such as clustering (grouping together similar documents), plus association rule mining (discovering rules that interrelate documents or terms within those documents). Especially if considering their extreme usefulness, with a few notable exceptions [19,11], taking advantage of these general techniques within digital library services does not appear to be widespread. The situation is similar in text mining processes within digital libraries, with very few examples of fully integrated services and workflows [48].

The computational expense to execute data or text mining based analysis has been identified as the major cause for the lack of more widespread use of such mining processes [37]. Only a very few natural language processing systems are available for general use, being sufficiently fast in serial mode to cope with even medium scale digital libraries without distributed computing. Further, the most advantageous method of integration into document processing workflows is often not obvious. To overcome this obstacle, [37] proposed that while the data grid integration is very important for dramatically increasing the scalability of digital library and preservation systems, the actual utility stems from the added integration of computationally expensive processing, extending the information available for discovery, analysis, and potential reuse.

The initial exploration of a distributed grid-based data and text mining system in a digital library context [37] considered support vector machines, naïve Bayes networks, and association rule mining, coupled with part-of-speech tagging and stemming. These processes were integrated with iRODS and Cheshire, requiring extensive changes of the latter. The feasibility of such integration is convincing, but appears cumbersome.

4 A flexible middleware for digital preservation in grids and clouds

This section describes our proposed middleware for digital preservation that is agnostic to whether the environment is a grid or a cloud. The middleware is in the Business Services layer of the SHAMAN core infrastructure, underlying most of the components that belong here. The middleware hides the complexity of the switch between a grid or a cloud, irrespective of whether the need for change arises from storage requirements or computational demand, enabling a smooth transition between the different types of infrastructures.

While it is the source of incessant debates of what a cloud actually is, first we outline a few working definitions that help limit the scope for the rest of the discussion (Subsection 4.1). Then we describe the various components of the proposed middleware, which is also illustrated in Figure 4. The first component is an archival framework that acts according to a predefined set of policies (Subsection 4.2). The remaining two sections address dynamic scalability of computations in grids and clouds for preservation workflows and additional support services (Subsections 4.3 and 4.4).

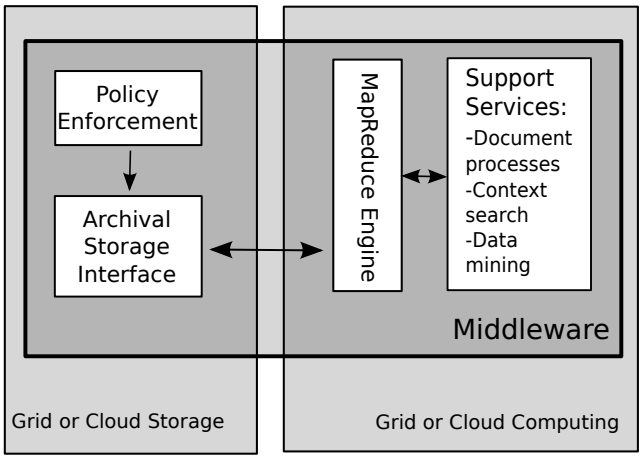

Fig. 4 Overview of the proposed framework

4.1 Grids, institutional and public clouds

By grid, we mean a software infrastructure that links distributed computational resources such as people, computers, sensors and data [16]. A data grid links distributed storage resources, from archival systems, to caches and databases, mapping data to a uniform logical name space to create global, persistent collections. Grids tend to be more loosely coupled, heterogeneous, and geographically dispersed than traditional clusters, in which the individual units are very similar and they are connected through a fast local area network.

The definition of a "cloud" is controversial. By a cloud, we simply mean a dynamically provisioned infrastructure or service [2]. Under this model, a user can dynamically provision any amount of computing resources from a (cloud) provider on demand and only pay for what is consumed [28]. Technically, this means that the user is paying for access to virtual machine instances that run a standard operating system. The virtualization technology enables the cloud provider to allocate available physical resources and enforce isolation between multiple users who may be sharing the same hardware. Once one or more virtual machine instances have been allocated, the user has full control over the resources and can use them for arbitrary computation. When the virtual instances are no longer needed, they

are destroyed, thereby freeing up physical resources that can be redirected to other users.

An institutional or private cloud refers to an internal data centre of a business or other organization that is not made available to the public. Private clouds offer the ability to host applications or virtual machines in a set of hosts owned by the company. The infrastructure has to be purchased and managed by the organization; in this sense, a private cloud is very similar to a grid [17]. One may argue that the level of abstraction is different, but the idea of dynamic provisioning of institutional resources is the same. Therefore we use the terms grid and institutional cloud interchangeably.

A public cloud makes a distributed infrastructure available in a pay-as-you-go manner to the public, the service being sold as utility computing. This is the term that describes cloud computing in the traditional mainstream sense. The idea behind utility computing is to treat computing resources as a metered service, like electricity or natural gas. Under this model, a user can dynamically provision any amount of computing resources from a (cloud) provider on demand and only pay for what is consumed [28]. Resource consumption is measured in machine-hours, breaking down to CPU-hours, bandwidth usage, etc.

We primarily view cloud computing as infrastructure-as-a-service, offering relatively bare bones systems on top of which a user or organization needs to deploy and manage their applications and data [36]. The biggest challenge in cloud computing is the lack of a standard or single architectural method which can meet the requirements of an enterprise cloud [35].

4.2 Archiving to a cloud

The iRODS system is already prepared to archive to a cloud. Its compound resource type makes it easier to integrate HPSS, Amazon S3, FTP and other types of resources into iRODS. A compound resource is a class where the POSIX data access type I/O calls such as open, read, write, lseek, close, etc are not readily available. Instead, it uses "put" and "get" calls to transfer entire files. The compound resource implementation in iRODS requires a cache class resource to be configured in the same resource group as the compound resource. Data stored in the compound resource cannot be accessed directly but through the cache resource using the put/get driver functions.

The S3 driver in iRODS was implemented based on the libs3 C library[2]. The current S3 protocol does not support some of the functionalities that are normally seen in file system type resources. Among the more important missing features are directory support and rename capability. The rename capability is particularly important since the iRODS server uses it quite often to move files and directories around such as moving files to the trash. Currently this functionality is emulated with a "copy" and "unlink" which naturally is not very efficient [43].

[2] `http://libs3.ischo.com/index.html`

4.3 Addressing the computational aspects of digital preservation

The steps of a preservation workflow tend to be computationally expensive. For instance, when migrating from one XML format to the other, one may insert an XSLT transformation in the pipeline. XSLT processors are increasingly optimized, using the kind of optimization techniques found in functional programming languages and database query languages. For example, static rewriting of the expression tree to move calculations out of loops, and lazy pipelined evaluation to reduce the use of memory for intermediate results, allow "early exit" when the processor can evaluate an expression without a complete evaluation of all subexpressions. There are also processors which use tree representations that are much more efficient than a general purpose Document Object Model (DOM) implementation, which is a non-optimized cross-platform and language-independent convention for representing and interacting with objects in an XML document. However, even with these optimizations, processing a single large document tree can take hours, provided that the computer has the necessary resources, particularly a satisfactory amount of physical memory.

XML transformations, while costly, are not the most expensive operations. Automatic metadata extraction may involve high-complexity natural language processing tasks such as deep parsing and named entity recognition. The use of these tasks can be prohibitive even for smaller collections due their computational requirements.

It is important to recognize the ad-hoc nature of these computations. Metadata extraction and migration are not frequently performed. Small organizations, or organizations that do not have a considerable budget for investments in information technology, do not have to maintain the resources permanently to deal with these operations. As many text mining applications prove, cloud computing is ideally situated for such situations.

IRODS offers a distributed execution framework via micro-services, however, these tasks are typically designed for computationally less demanding applications,

such as the simple parsing of a document. In a resilient distributed infrastructure, a more robust method is necessary. MapReduce was developed by Google to this end in the early 2000s, and the framework was first published in 2004 [12]. The publication spawned several open source efforts to implement the framework. Hadoop, now a top-level Apache project, was first released in 2006, and it has become the most popular such open source implementation [45].

MapReduce focuses on scaling out, using a large number of low-cost commodity servers instead of a small number of expensive HPCs. It assumes that failures are common: a failure is not the exception, but the norm. Redundancy is a requirement. Exploiting the resources in the cloud can be problematic for DP, as it requires persistence and high reliability [39]. In this context, the MapReduce framework helps. DP in a traditional grid environment suffers from a number of threats. These threats include component failures such as media faults, hardware faults, software faults, communication faults, network services failures. These faults are assumed as part of the normal operation in a MapReduce framework, and are addressed at lower level, application developers do not have to deal with them directly. Redundancy is built-in, with a computation launched at least three nodes simultanously, and if one result is different than the ones on the other two nodes, the subtask is executed again. Debugging, however, can be a major issue. Given the complexity of the task, a gradual scaling out can help to identify errors. A task could be launched locally or in a local pseudo-cluster, then in a single-node cloud instance, then a full-scale launch may follow. However, if the input collection is inconsistent, finding out how the problem persists in the output collection is an unresolved challenge.

The basic data structure of MapReduce is key-value pairs. Keys and values can be arbitrarily complex data structures. For instance, for a collection of web pages, the keys can be the URLs and the values are the HTML content. For a graph, keys can be the node identifiers, while values are the adjacency lists of those nodes. The output of the mapper is a sorted list of key-value pairs. The mapper also commonly performs input format parsing, projection (selecting the relevant fields), and filtering (removing records that are not of interest).

In the reduce step, the master node then takes the answers to all the sub-problems and combines them in a way to get the output - the answer to the problem it was originally trying to solve. As the processing gets more complex, this complexity is generally manifested by having more MapReduce jobs, rather than having more complex map and reduce functions. In other words, a developer thinks about adding more jobs, rather than increasing the complexity of the jobs.

The MapReduce framework, while hiding the complexity of the underlying architecture, forces the user to think in terms of map and reduce operations. When it comes to document processing, it is difficult to divide a single document to smaller pieces. The unit for the map operation is therefore a single XML document [47]. This approach integrates well with document pipelines and document validation checkpoints modelled by a preservation workflow which allows one to define and design document processes while producing an abstract representation that is independent of the implementation. A workflow designed for a document is invoked from the map routine of a MapReduce job, with the identity operator acting as the reduce function. The output is the transformed document which then may undergo further processing in subsequent MapReduce jobs.

To implement the above approach, we choose Apache Hadoop, which is a software framework that supports data-intensive distributed applications. Hadoop is essentially a MapReduce implementation (i.e., the MapReduce engine) combined with a distributed file system (HDFS). We were not measuring the scalability of the software for distributed computing, hence we opted for Hadoop as stable and widely used tool, and did not consider other options.

The output of an XML processing pipeline is a collection of XML documents in METS format, as mentioned in Section 3.2. Going beyond DP, further automatic content extraction may be performed at document or collection level. Such further processing may be used to support digital curation (see also Section 4.4). Indexing and machine learning algorithms are likely to require a different XML input format. Interfacing between the document processing pipeline and further steps can be done with simple XSLT transformations.

4.4 Additional services to facilitate access and reuse

As outlined in section 3.3, full-text indexing and retrieval are essential components to enable future access of digital collections. Cheshire uses the Parallel Virtual Machine to scale to a distributed environment, which limits its flexibility and may not work in a cloud environment where latencies can be haphazard. Therefore we suggest using Nutch [8], which relies on the MapReduce framework of Hadoop to index files with Lucene [18].

The parallel indexing operation in the MapReduce model works as follows. The data to be indexed is partitioned into segments of approximately equal size. Each

segment is processed by a mapper task that generates the key-value pairs for that particular segment, where the key is an index term and the value is the set of documents that contain that term. The value may be coupled with additional information, such as the location of the term in the document. In the reduce phase, each reducer task collects all the pairs for a given key, thus producing a single index table for that particular term. Once all the keys are processed, we have the entire inverted index for the entire collection. Nutch is particularly well suited for scaling out with a large number of commodity hardware [30], which is the exact situation with cloud resources.

As pointed out in section 3.4, another aspect of the DP area is digital curation which is being increasingly used for the actions needed to add value to and maintain these digital assets over time for current and future generations of users [6]. Digital curation naturally involves the preservation of collections, and also entails the semantic and ontological continuity and comparability of the collection content. Document and collection-level metadata are crucial to support the goals of digital curation. We anticipate that in the tested framework, machine learning algorithms can be useful to support further metadata extraction. This would involve indexing the outcome of a document processing pipeline, either the full-text documents or the extracted features. Once indexing is performed, various clustering, classification, and modelling algorithms can be deployed to add metadata to individual documents or to the entire collection.

The natural language processing tools can be integrated with the MapReduce framework in a similar fashion to the preservation workflow. Moving beyond individual documents, a wide range of machine learning libraries has already been developed for Hadoop, such as the ones included in Mahout [31], a scalable machine learning library based on MapReduce. This makes it easy to develop or extend flexible and transparent data mining methods for DP.

5 Discussion of the results

5.1 Implementation

The implementation relies on the decoupling of the MapReduce framework from the XML processing pipeline, and since there are no internal dependencies for the processes, the workload is naturally parallel. A process designed in Xeproc is exported via the EMF interface to Python, and is executed on individual documents that are mapped out to the nodes in the cloud by a relatively simple MapReduce driver.

For local experiments, we used a workstation with 24 GB of main memory, one quad-core Intel Xeon E5620 CPU with two logical units in each core and 2 TB of storage, running in a 64-bit environment. For cloud computations, we used Amazon Web Services (AWS). It is possible to run Hadoop on Amazon Elastic Compute Cloud (EC2) and Amazon Simple Storage Service (S3). While Amazon Web Services is undoubtedly the market leader in cloud computing, other providers, including at least one open source solution, exist. We decided in favor of AWS due to the maturity of its products.

The Python module requires bootstrapping across all instances in the cluster. This involves copying the Xeproc player from S3 to the local drive, then installing modules needed to run it.

We use two types of Amazon EC2 instances: small standard instances (m1.small) and large standard instances (m1.large). The former consists of 1.7 GB of main memory, one EC2 Compute Unit (one virtual core with one EC2 Compute Unit), 160 GB instance storage (150 GB plus 10 GB root partition), and the software architecture is 32-bit. A large instance includes 7.5 GB of main memory, four EC2 Compute Units (two virtual cores with two EC2 Compute Units each), 850 GB instance storage, and the platform is 64-bit.

5.2 Document processing pipeline

Demonstrating the capabilities of Xeproc, we use a process which recognizes and extracts the table of contents of a document [14, 13]. This pipeline is composed of the following steps (see also Figure 3):

- Entry-level PDF to XML converter[3] for PDF files;
- Ad-hoc XSLT transformations for XML files;
- Page header and footer recognition;
- Text reading order reconstruction and paragraph segmentation;
- Caption detection;
- Table of contents analysis.

The first steps consist of recognizing and deleting document elements which can introduce noise during the table of contents analysis: page headers and footers are identified and deleted in order to reduce noise (running titles); so are captions to eliminate tables of figures as potential table of contents. The text reading order and paragraph segmentation step generates a proper content flow which is used by the table of contents component. Finally the table of contents is extracted, and titles in the document body are marked up as heading elements (Figure 5).

[3] `http://sourceforge.net/projects/pdf2xml/`

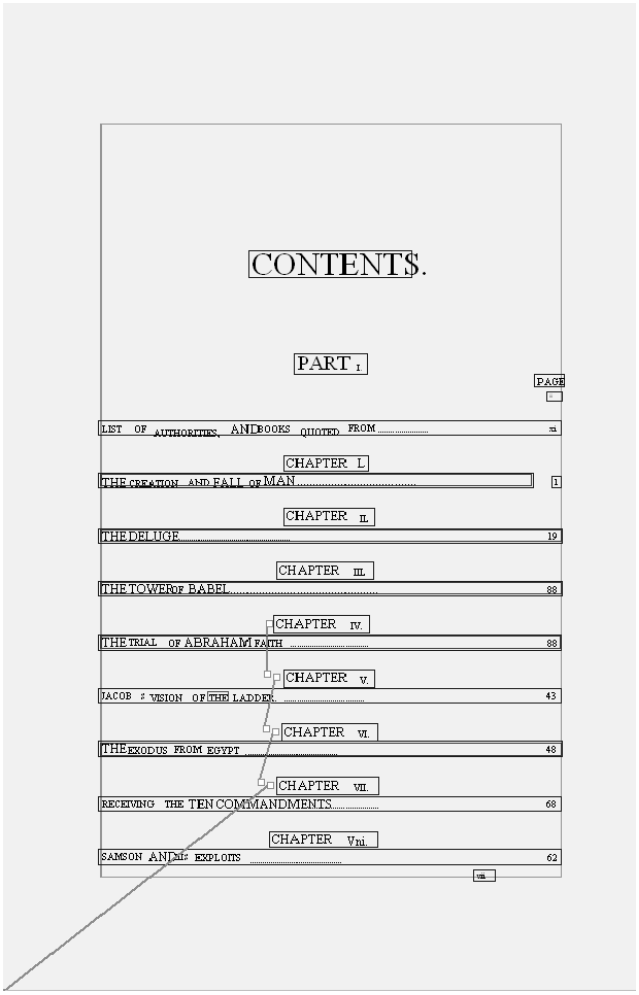

Fig. 5 An example of an identified table of contents in a PDF file [13]

Validation checkpoints are associated with each step. One type of validation aims at validating the step output against an XML Schema. Others perform more specific validations triggering errors or warnings and using XSL Transformations (especially XSLT 2.0) or ad hoc engines. For instance, an XSLT 2.0 validation detects overlapping paragraphs after the paragraph segmentation step and triggers a warning which points out some difficult or unexpected content layout. The user interface allows for visual inspection of the list of errors and warnings.

5.3 The Dataset

The collection being studied is a collection of doctoral theses hosted by the German National Library[4]. At the point of downloading the collection, it contained 94,437 theses. The total volume of the PDF files is over 500GB.

The accompanying metadata is in Dublin Core format, which is a set of elements providing a small and fundamental group of text elements using only fifteen fields by which most resources can be described and cataloged.

The collection is multilingual with approximately 75 % of it being in German, and over 20 % in English. The presence of other languages is almost negligible.

[4] `http://deposit.d-nb.de/index_e.htm`

5.4 The overhead of using a MapReduce framework

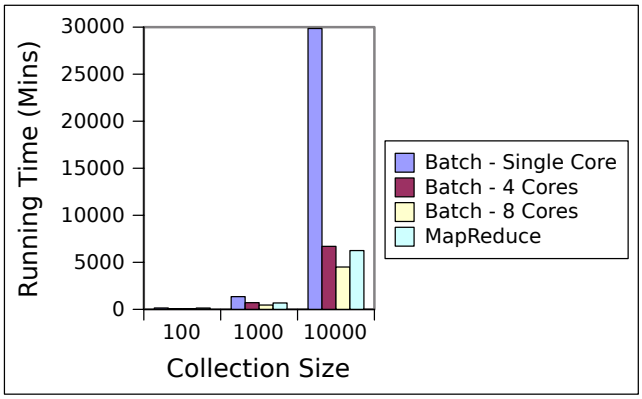

Fig. 6 Comparison of running times on a single local node

On a small subset of a hundred documents, MapReduce performance is close to single-core batch processing (Figure 6). This is probably due to the overhead of setting up the framework. For larger collection sizes, MapReduce traces the quad-core performance. This is expected, since we see three to four map tasks running simultaneously.

Load balancing is not optimized for the batch processor, as it simply divides up the collection to tasks of approximately equal size. The size, however, does not correspond well with the actual time spent on computation. This gives a slight advantage to MapReduce, since it has a sophisticated mechanism for load balancing. This is not critical when comparing with the quad-core performance, as the differences between the finishing times of individual threads are marginal for four concurrent threads. The differences in running eight processes are more significant, the slowest thread finishes 1.6 times slower than the fastest. Hence the eight-thread performance is not directly comparable with that of MapReduce.

5.5 Running time in the cloud

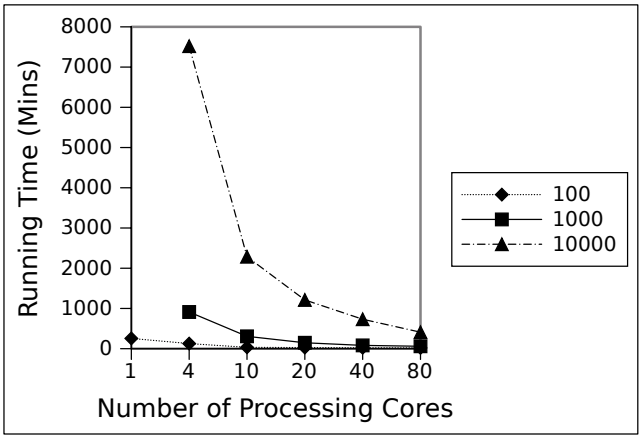

Fig. 7 Comparison of running times with different collection sizes

Before we compare the running times of executing a pipeline locally versus in the cloud, we must note some changes that are required for scaling out. The default configuration for the EC2 m1.small and m1.large instances has 1.7 GB main memory with 900 MB swap, and 7.5 GB main memory with no swap, respectively. As the memory requirements on the single local node hinted, this is not sufficient for running several processes simultaneously. An m1.small instance runs two child processes for MapReduce tasks, and an m1.large instance launches four. To accommodate the occasional peak memory usage, 4 GB of swap memory was configured for both types of instances. We observed a maximum of 2.1 GB swap memory used, which is a surprising result, since local single-node runs saw up to 8 GB of memory usage for a single Python process.

The document processing pipeline is also demanding on the CPU, not only on the memory. Since the Python interface does not comply with the stream interface of Hadoop, a map task which spawns the Python process may not be able to send "heartbeat" signals for long periods of time. These heartbeat signals are required by the framework, and if they are not received periodically, the task is considered failed. To avoid this, we removed the need for these signals. As the Python module always terminates, a permanent lock-up will not occur. It may happen that several pipelines require long computations simultaneously – a larger number of nodes will balance the load better.

The times shown in Figure 7 do not include the time needed to launch an instance or a cluster. This depends on the demand, the type of instances launched, and the number of instances requested. For a cluster of m1.large instances the starting time alone can be ten minutes or more. We focus our attention on the actual execution time of the document processes. The time and cost of moving data in and out of the cloud is negligible compared to the exectution time.

The closest equivalent to our local node is a single m1.large instance. The running times are marginally better on the local node for collection size above a hundred documents (75 % and 86 % of the running time compared to the single quad-core cloud instance). This is in line with our intuition, as the local node has four times more main memory and does not have to rely on swap memory at all. The Hadoop configuration is more fine-tuned to virtual instances, and that may cause the better performance.

On small collections, there is little to gain by scaling to a high number of nodes or processing cores. As the collection size increases, it makes more and more sense to use a larger cluster. With a twenty-instance cluster of m1.large nodes, the running time drops from two days to just seven hours. Cost, however, becomes an important aspect in choosing the correct cluster size.

5.6 Cost analysis of computing and storage

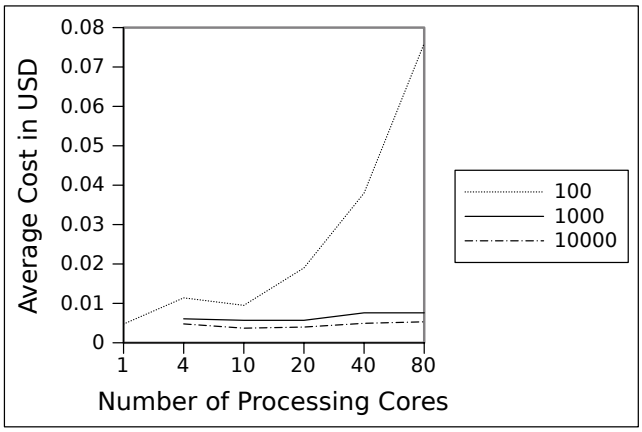

Fig. 8 Comparison of average cost of computations with different collection sizes

The price analysis is based on instance pricing in the EU West region (Ireland) as of September 2009. The prices indicated do not include VAT. The pricing for compute instances does not include a small fixed charge for using the MapReduce framework, indicating the cost of using EC2 instances only.

Striking an optimal average price depends largely on the collection size. Since full hours have to be paid for partial hours as well, small data sets are the cheapest to

process on a single or a few m1.small instances (Figure 8).

The scenario is different for larger collections. They tend to have a minimum average cost in the middle range. For a thousand documents, the lowest cost is US$0.0057 per document in clusters of ten or twenty instances. For ten thousand documents, the lowest average price is US$0.0037 with a cluster of ten m1.small instances. This also hints at a trend the larger the collection the lower the average cost per processed object.

Beyond the compute instances, we also look at persistent storage in the cloud as an alternative to local hosting. Amazon S3 charges US$0.150 per gigabyte per month for the first fifty terabytes of data. To store the full collection, it would amount to approximately US$80.00 per month. This, however, is not the total cost of storage, as the access requests also cost US$0.01 per thousand. Depending on the purpose of the collection, this may prove to be a substantial sum. Public access to the collection stored in S3 would also need third-party solutions, involving service level agreements, further increasing the cost.

6 Organizational issues

The cloud, like the grid, is another variant of distributed computing. A valuable comparison between the cloud and grid has already been carried out by [7]. Rather than perform another comparison we feel it more to the point to try to highlight the factors that need to be considered when making use of third-party services. These are as follows:

- The environment in which the application acts on the data must be recorded. An application that extracts metadata information or transforms data from one format to another may behave differently on different platforms. Although the results may be within acceptable tolerance it will still be important to capture the environment (such as the operating system, the version of the application, etc) to help users of the data to assess whether observed features are genuine or an artifact of the processing.
- There is also the critical need to validate each result from the cloud. To ensure it conforms to the expected result e.g. reference samples may need to be run in order to understand that the results are within tolerance. This is because the preservation system has no control over the cloud resources which may change from one day to the next and may introduce a subtle bug that could not be detectible if there is no validation based on significant properties of the data using a reference sample.
- An SLA on the quality and level of service must be agreed. An organisation tasked with digital preservation will have defined quality of service measures and will need to ensure the third-party service meets or exceeds those measures. A service level agreement would contain those quality of service measures as well as a means to verify they are met. The quality of service agreement can range from security through service reliability to performance. The SLA should also detail how data are to be removed or deleted in the case of termination of the contract.
- Many cloud providers offer attractive SLAs. An SLA may specify the levels of availability, serviceability, performance, operation, or other attributes of the service. The "level of service" can also be specified as "target" and "minimum", which allows users to be informed what to expect. In the context of digital curation, if the collection is preserved in the cloud, the SLA has to explicitly define the persistency of storage. Cloud providers may have fundamentally different SLAs, which makes their comparison difficult. An ongoing European Union funded FP7 project, SLA@SOI, is researching aspects of multi-level and multi-provider SLAs within service-oriented infrastructure and cloud computing [9,41]. The project will eventually deliver predictability, dependability, and transparency in SLA management aimed at making it easier to choose a cloud provider suitable for DP purposes [29].
- As suggested above, beyond the physical threats of data loss, other challenges in a grid environment include organizational issues, such as management failures and economic failures. While these can be considered at a higher organizational level through proper preservation policies, with a cloud provider involved, these policies can be more difficult to enforce. Economic failures are also a threat to cloud providers, hence a single provider might prove to be hazardous for preservation.

7 Conclusion

With a paradigm shift in the making toward a service-oriented architecture, DP is one of the areas to benefit from this change. Considerations suggest that especially small organizations should welcome the turn as an attractive upcoming solution to their related problems. To test the feasibility of this assumption, we built a preservation workflow in a cloud processing environment to show that the process is smoothly running.

The competitive aspects of the new technology, including its cost assessments, are too early to address

but as shown by parallel experiments on a high performance workstation, memory and CPU requirements of DP are quite considerable and may be beyond reach for several institutions. As first results by cloud computing suggest, even a single node m1.large instance has a performance close to that of a decent workstation which makes running workflows possible for everyone. Moreover, a single workstation may be more expensive to rent for a longer term than running the same calculations on a cluster, as shown in the average cost diagrams. By picking the cloud configuration with the lowest estimated average cost, bulky DP jobs can be done at an affordable price and with flexibility beyond that of fixed resources.

We believe that the architecture outlined in this paper advances the state-of-the-art in DP for the following reasons:

- The procurement of an expensive server or a grid can be replaced by service level agreements with the cloud provider;
- The flexibility is unprecedented in terms of scale and document process design;
- Ad-hoc peak computations that are typical in document processes are easily addressed;
- Persistent storage in the cloud is a viable alternative to local servers;
- The MapReduce framework enables an easy integration of various support services of DP, such as document migration, metadata extraction, natural language processing, full-text indexing and retrieval, and data mining.

With the emergence of high-performance computing instances in the cloud, computations on a massive scale have become available to technically every organization. Our future work includes further investigation into this emergent field with implications for digital libraries and digital preservation [46]. We also hope to extend the scope to alternative computing infrastructure, such as graphics hardware, which would be able to accelerate language technology services with speedups unseen before.

Acknowledgements We are grateful to Thierry Jacquin, Jean-Pierre Chanod, and Hervé Déjean at the Xerox Research Centre Europe, Grenoble for their help with document processes, and to Adil Hasan at the University of Liverpool for discussions on grid-based solutions. We would also like thank our collaborators at the University of Liverpool for their help in developing the original grid-based iRODS implementation.

References

1. Allinson, J.: OAIS as a reference model for repositories. Tech. rep., UKOLN, University of Bath (2006)
2. Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I.: Above the clouds: A Berkeley view of cloud computing. Tech. rep., EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28 (2009)
3. Ball, A.: Briefing paper – the OAIS reference model. Tech. rep., UKOLN, University of Bath (2006)
4. Barateiro, J., Antunes, G., Borbinha, J., Lisboa, P.: Addressing digital preservation: Proposals for new perspectives. In: Proceedings of InDP-09, 1st International Workshop on Innovation in Digital Preservation. Austin, TX, USA (2009)
5. Barateiro, J., Antunes, G., Cabral, M., Borbinha, J., Rodrigues, R.: Using a grid for digital preservation. In: Proceedings of ICADL-08, 11th International Conference on Asian Digital Libraries: Universal and Ubiquitous Access to Information, pp. 225–235. Kuta, Indonesia (2008)
6. Beagrie, N.: Digital curation for science, digital libraries, and individuals. International Journal of Digital Curation **1**(1), 3–16 (2006)
7. Bégin, M., Jones, B., Casey, J., Laure, E., Grey, F., Loomis, C., Kubli, R.: An EGEE comparative study: Grids and clouds – evolution or revolution. Tech. rep., Enabling Grids for E-sciencE-II (EGEE-II) Project Report INFSO-RI-031688 (2008)
8. Cafarella, M., Cutting, D.: Building Nutch: Open source search. Queue **2**(2), 54–61 (2004)
9. Comuzzi, M., Kotsokalis, C., Spanoudakis, G., Yahyapour, R.: Establishing and monitoring SLAs in complex service based systems. In: Proceedings of ICWS-09, 7th International Conference on Web Services, pp. 783–790. Los Angeles, CA, USA (2009)
10. Cundiff, M.: An introduction to the Metadata Encoding and Transmission Standard (METS). Library Hi Tech **22**(1), 52–64 (2004)
11. Darányi, S., Wittek, P., Dobreva, M.: Using wavelet analysis for text categorization in digital libraries: a first experiment with Strathprints. International Journal on Digital Libraries (2011)
12. Dean, J., Ghemawat, S.: MapReduce: Simplified data processing on large clusters. In: Proceedings of OSDI-04, 6th International Symposium on Operating Systems Design & Implementation. San Francisco, CA, USA (2004)
13. Déjean, H.: Numbered sequence detection in documents. Document Recognition and Retrieval XVII **7534**(1), 753,405–12 (2010)
14. Déjean, H., Meunier, J.L.: On tables of contents and how to recognize them. International journal on document analysis and recognition **12**(1), 1–20 (2009)
15. Engel, F., Klas, C., Brocks, H., Kranstedt, A., Jäschke, G., Hemmje, M.: Towards supporting context-oriented information retrieval in a scientific-archive based information lifecycle. In: Proceedings of Cultural Heritage online. Empowering users: an active role for user communities, pp. 135–140. Florence, Italy (2009)
16. Foster, I., Kesselman, C.: The grid: blueprint for a new computing infrastructure. Morgan Kaufmann (2004)
17. Foster, I., Zhao, Y., Raicu, I., Lu, S.: Cloud computing and grid computing 360-degree compared. In: Proceedings of GCE-08, Grid Computing Environments Workshop, pp. 1–10 (2008)
18. Gospodnetic, O., Hatcher, E., et al.: Lucene in Action. Manning (2005)
19. Han, H., Giles, C., Manavoglu, E., Zha, H., Zhang, Z., Fox, E.: Automatic document metadata extraction using support vector machines. In: Proceedings of JCDL-

03, 3rd ACM/IEEE-CS Joint Conference on Digital Libraries, pp. 37–48. Houston, TX, USA (2003)
20. Hedges, M., Blanke, T., Hasan, A.: Rule-based curation and preservation of data: A data grid approach using iRODS. Future Generation Computer Systems **25**(4), 446–452 (2009)
21. Hedges, M., Hasan, A., Blanke, T.: Management and preservation of research data with iRODS. In: P. Mitra, C. Giles, L. Carr (eds.) Proceedings of CIKM-07, 1st Workshop on CyberInfrastructure: Information Management in eScience, in conjuction with 16th Conference on Information and Knowledge Management, pp. 17–22. Lisbon, Portugal (2007)
22. Innocenti, P., Ross, S., Maceciuvite, E., Wilson, T., Ludwig, J., Pempe, W.: Assessing digital preservation frameworks: the approach of the SHAMAN project. In: Proceedings of MEDES-09, 1st International Conference on Management of Emergent Digital EcoSystems, pp. 412–416. Lyon, France (2009)
23. ISO 14721: Reference model for an Open Archival Information System (OAIS) fCCSDS 650.0-B-1 Blue book (2003)
24. Jacquin, T., Déjean, H., Chanod, J.P.: Xeproc©: A model-based approach towards document process preservation. In: M. Lalmas, J. Jose, A. Rauber, F. Sebastiani, I. Frommholz (eds.) Research and Advanced Technology for Digital Libraries, *Lecture Notes in Computer Science*, vol. 6273, pp. 538–541 (2010)
25. Knight, G., Hedges, M.: Modelling OAIS compliance for disaggregated preservation services. International Journal of Digital Curation **2**(1), 62–72 (2008)
26. Larson, R., Sanderson, R.: Grid-based digital libraries: Cheshire3 and distributed retrieval. In: Proceedings of JCDL-05, 5th Joint Conference on Digital Libraries, pp. 112–113. Denver, CO, USA (2005)
27. Larson, R., Sanderson, R.: Cheshire3: retrieving from tera-scale grid-based digital libraries. In: Proceedings of SIGIR-06, 29th Annual International Conference on Research and Development in Information Retrieval, pp. 730–730. Seattle, WA, USA (2006)
28. Lin, J., Dyer, C.: Data-Intensive Text Processing with MapReduce. Morgan & Claypool (2010)
29. Metsch, T., Edmonds, A., Bayon, V.: Using cloud standards for interoperability of cloud frameworks. Tech. rep., SLA@SOI (2010)
30. Michael, M., Moreira, J., Shiloach, D., Wisniewski, R.: Scale-up x scale-out: A case study using Nutch/Lucene. In: Proceedings of IPDPS-07, 21st International Parallel and Distributed Processing Symposium, pp. 1–8. Long Beach, CA, USA (2007)
31. Owen, S., Anil, R., Dunning, T., Friedman, E.: Mahout in Action. Manning Publications Co (2010)
32. Phelps, T., Watry, P.: A no-compromises architecture for digital document preservation. Research and Advanced Technology for Digital Libraries pp. 266–277 (2005)
33. Phelps, T., Wilensky, R.: The multivalent browser: a platform for new ideas. In: Proceedings of DocEng-01, 1st Symposium on Document Engineering, pp. 58–67. Atlanta, GA, USA (2001)
34. Rajasekar, A., Moore, R., Hou, C., Lee, C., Marciano, R., de Torcy, A., Wan, M., Schroeder, W., Chen, S., Gilbert, L., et al.: iRODS primer: Integrated rule-oriented data system. Synthesis Lectures on Information Concepts, Retrieval, and Services **2**(1), 1–143 (2010)
35. Rimal, B., Jukan, A., Katsaros, D., Goeleven, Y.: Architectural requirements for cloud computing systems: An enterprise cloud approach. Journal of Grid Computing **9**(1), 3–26 (2011)
36. Rings, T., Caryer, G., Gallop, J., Grabowski, J., Kovacikova, T., Schulz, S., Stokes-Rees, I.: Grid and cloud computing: opportunities for integration with the next generation network. Journal of Grid Computing **7**(3), 375–393 (2009)
37. Sanderson, R., Watry, P.: Integrating data and text mining processes for digital library applications. In: Proceedings of JCDL-07, 7th ACM/IEEE-CS Joint Conference on Digital Libraries, pp. 73–79. Vancouver, Canada (2007)
38. SHAMAN Consortium: WP2.D2.3 Specification of the SHAMAN reference architecture. Tech. rep., SHAMAN (2009)
39. Skinner, K., Schultz, M.: A Guide to Distributed Digital Preservation. Educopia Institute (2010)
40. Sunderam, V.: PVM: A framework for parallel distributed computing. Concurrency: practice and experience **2**(4), 315–339 (1990)
41. Theilmann, W., Yahyapou, R.: SLA@SOI – SLAs empowering a dependable service economy. ERCIM News **83** (2010)
42. Tidwell, D.: XSLT: Mastering XML Transformations. O'Reilly Media, Inc. (2007)
43. Wan, M., Moore, R., Rajasekar, A.: Integration of cloud storage with data grids. In: Proceedings of ICVCI-09, 3rd International Conference on the Virtual Computing Initiative. Research Triangle Park, NC, USA (2009)
44. Watry, P.: Digital preservation theory and application: Transcontinental persistent archives testbed activity. International Journal of Digital Curation **2**(2), 41–68 (2007)
45. White, T.: Hadoop: The Definitive Guide. O'Reilly Media (2009)
46. Wittek, P., Darányi, S.: Leveraging on high-performance computing and cloud technologies in digital libraries: A case study. In: Proceedings of HPCCloud-11, Workshop on Integration and Application of Cloud Computing to High Performance Computing, pp. 606–611. Athens, Greece (2011)
47. Wittek, P., Jacquin, T., Déjean, H., Chanod, J.P., Darányi, S.: Xml processing in the cloud: Large-scale digital preservation in small institutions. In: Proceedings of DataCloud-11, 1st International Workshop on Data Intensive Computing in the Clouds in conjunction with the 25th IEEE International Parallel and Distributed Computing Symposium. Anchorage, AK, USA (2011)
48. Witten, I., Don, K., Dewsnip, M., Tablan, V.: Text mining in a digital library. International Journal on Digital Libraries **4**(1), 56–59 (2004)