ToMaR - A Data Generator for Large Volumes of Content

Rainer Schmidt
AIT Austrian Institute of Technology
rainer.schmidt@ait.ac.at

Matthias Rella
AIT Austrian Institute of Technology
matthias.rella.fl@ait.ac.at

Sven Schlarb
Austrian National Library
sven.schlarb@onb.ac.at

***Abstract*—We present ToMaR, a scalable application that supports the efficient integration of legacy applications within a MapReduce environment. The work is motivated by scenarios for scalable content processing developed in the context of the EC project SCAPE. ToMaR specifically addresses the need for extracting data sets from large volumes of binary content based on existing, content-specific applications within a scalable data management environment. This paper discusses the main functionalities of ToMaR and describes how ToMaR is utilized as part of a typical workflow. We present a real-word scenario that makes use of ToMaR for the characterization of archived web content. A workflow and experimental results which have been produced using sample content from the Web Archive Austria are discussed.**

I. Introduction

ToMaR has been developed in the context of the Scalable Preservation Environments project (SCAPE)[1] to support the processing of large volumes of content available in digital libraries and archives. Data sets that are being analyzed include millions of high-resolution images from digitized books, many hundreds of thousands of individual data items generated by scientific instruments, and tens of TBs of web-content containing hundreds of millions of resources. Libraries and archives are obliged to curate these data sets in order to manage and preserve them allowing users to access, reuse, or analyze them.

An important activity when dealing with the management of archived content is the extraction of metadata from the ingested items enabling the controlled execution of subsequent analysis and/or data management actions. Archival information systems [1] are traditionally handling data extraction and manipulation activities through batch pipelines provided by data curation software [2] during the ingest process. While this approach is being commonly used in conventional data archiving systems, it is not applicable when processing very large volumes of content, like those mentioned above.

With the advent of Internet and cloud computing, new technologies have been developed which enable us to process data at a very large scale [3]. While this technology shift has become natural for Internet companies who are driving this development, it imposes - besides new opportunities - significant problems for traditional content holders as well as for the domain scientists aiming to retrieve new information from the archived content.

Scale-out architectures have proven to work well for storing and analyzing large volumes of unstructured alphanumeric data sets. This is supported by frameworks that enable a direct mapping of data management techniques (like data base operations) to massively parallel architectures [4], allowing us to perform well known data analytics tasks on large-scale data sets by using implementations like Apache Pig [5] or the Mahout [6] library. Decomposing and processing data BLOBS like audiovisual content in such an environment is possible but much more difficult to generalize [7].

SCAPE is developing a platform and corresponding tool kit that utilizes data-intensive technologies for analyzing and preserving digital content. Our prototype implementation replaces different key components of the archival system with scalable solutions. Integration with the presently developed Fedora 4 repository [8] provides a horizontally scalable digital object management system supporting scalable digital object management and large-scale access/ingest functionality. Storage, automated data-base building, and data analytics have been built on top of the Apache Hadoop [9] framework. A key requirement for the system, however, is support for legacy applications which are mainly required for generating data sets from the initial binary content. ToMaR, a tool which is being heavily used by content providers in the SCAPE project, implements this functionality.

Scalable environments like Hadoop require the user to conform to the MapReduce programming model, take advantage of a higher-level language (like Pig or Hive), or make use of utilities like the Hadoop streaming API. While these methods provide key abstractions for processing large-scale data sets, they are not directly applicable to the processing of binary content for a number of reasons, as explained later in this paper. ToMaR provides a flexible tool allowing practitioners to easily execute sequential applications efficiently in a MapReduce environment. This enables one to perform typical data curation actions (like file format identification, data validation, or the extraction of metadata) mostly requiring the use of legacy applications with minimal effort. By relying on Hadoop as its execution environment, ToMaR provides a robust pre-processing application, allowing one to generate large-scale data sets in a way that makes

[1] http://scape-project.eu

them directly usable through a variety of data management frameworks present in the Hadoop eco-system.

II. Motivation

The development of ToMaR was motivated by the need for an application that supports the execution of legacy applications in a MapReduce environment, as employed by the SCAPE Preservation Platform [10]. The SCAPE platform developed in the context of the Scalable Preservation Environments (SCAPE) project provides an infrastructure that targets the scalability of preservation environments used by digital libraries and archives. The goal is to enhance the scalability of storage capacity and throughput of digital object management systems based on varying the number of computer nodes available in the system and to support the execution of large-scale data analytics tasks against different sets of digital content. The SCAPE Testbeds implement workflows that utilize real world data from three different application areas: Digital Repositories from the library community, Web Content from the web archiving community, and Research Data Sets from the scientific community.

The SCAPE platform prototype implementation is based on existing, mature software components like Apache Hadoop, the Taverna Workflow Management Suite, or the Fedora Digital Asset Management System, and implements a set of additional services on top of these software components to specifically support scalability and integration with digital preservation processes as well as to integrate with other SCAPE components. Figure 1 provides an overview of the main entities of the SCAPE preservation platform and shows their interactions.

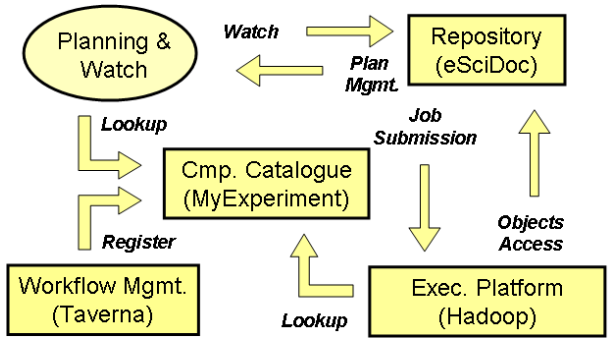

Figure 1. Components and services of the SCAPE Preservation Platform. The available software components provide support for workflow design and description, registration and lookup of preservation components, storage and analytics, and digital object management and efficient access. Integration with the SCAPE Preservation Planning and Watch components is supported through the Component Catalogue Lookup API and the Repository Plan Management and Watch APIs.

A key challenge of the platform was to efficiently support the integration of legacy tools most scenarios rely on, within its parallel execution environment. One example is the management of scientific data coming from neutron, x-ray, or muon instruments like those provided by the UK Science and Technology Facilities Council[2]. Many of these data sets produced by large scale scientific facilities have been stored in custom, locally developed data formats like RAW. To curate, analyze, and preserve hundreds of thousands of these data sets, it is required to transform them to NeXus, which is a common Proton and Neutron Community format. Converters from Raw-to-NeXus exist as well as tools for NeXus validation and metadata generation. Although large-scale data management techniques are required to cope with these data sets, it is not feasible to re-engineer the individual applications to directly support RAW file format interpretation on a MapReduce platform like Hadoop.

III. How ToMaR Works

ToMaR[3] is essentially a flexible application wrapper that supports the execution of sequential tools and APIs on Hadoop-based MapReduce clusters. The goal of ToMaR is to enable the efficient execution of a variety of legacy applications based on a simple user-supplied xml descriptor. This includes handling ad-hoc scripts, pre-installed application packages, workflows, file-based and streaming IO, APIs, and command pipes. Another goal is the integration of the external processes with the MapReduce model and data IO with Hadoop-specific concepts like HDFS, HBASE, or SequenceFiles. ToMaR must also be able to handle short running processes against large amounts of files efficiently. A key concern here is to avoid significant performance degradation being introduced by startup overheads of the map and reduce tasks.

A. Application Shipping

ToMaR does not handle the shipping of legacy applications to the computational nodes of a cluster. Instead, it assumes that the required applications are made available via the Hadoop framework or the node's operating system. Hadoop provides built-in support for shipping Java-based applications to the computational nodes via its distributed cache. However, many use-cases found in the content management domain require the employment of applications that rely on software that must be pre-installed on the operating system. Examples are multimedia codecs, image processing libraries, or characterization tools. On the SCAPE platform, the automated deployment of software is handled through the Debian package management system. Other techniques allowing us to configure a large cluster on demand include FAI and PXE. The SCAPE development infrastructure provides a Fully Automated Installation (FAI) server[4] for setting up the cluster nodes. The service allows us to easily add and fully configure new nodes to the system, which

[2]http://www.openplanetsfoundation.org/blogs/2012-07-10-research-datasets-testbed-stfc-data-what-and-why

[3]available at: https://github.com/openplanets/tomar

[4]available at: http://fai-project.org/

```
<operation name="sisHTML">
  <command>
    if [ "$( file - | awk "{print \$2}" )" == HTML ];
    then echo "HTML" ; fi
  </command>
  <inputs>
    <stdin>true</stdin>
  </inputs>
</operation>
```

Figure 2. Fragment of a tool specification document that implements an ad-hock script using the Linux file utility to identify HTML files. Input is read from the standard input stream.

can be booted via a network card using PXE, a pre-boot execution environment most modern network cards support. Additionally, the Eucalyptus cloud stack [11] has been used to deploy individually configured clusters on demand. Worker nodes of the infrastructure run the XEN hypervisor and a Debian distribution that includes a XEN Dom0 kernel.

B. Tool Specification Language

The SCAPE Tool Specification Language provides a simple XML schema[5] to formalize the usage of different software packages (tools) by specifying properties like API calls, configuration parameters, or defining how IO is handled. Tool specification documents (toolspecs) are XML documents that define a set of operations that can be carried out by the *tool* it defines. Tool specification documents can describe general patterns for using a single software package, or define new operations, e.g. for a complex command-line invocation. The term *tool* however refers to any possible combination of executable software packages (like scripts, APIs, or piped commands), which can be defined by a tool specification document. Important concepts of the toolspec schema are `<operation>` defining an new operation, `<command>` defining an command-line or API call, `<input>/<output>` defining IO properties like required inputs, generated outputs, file/stream-based IO. Figure 2 provides a simple example for expressing a Linux shell command sequence as a toolspec operation. Data is read from standard input and results are sent to standard output using the Linux *echo* command. ToMaR can be used to execute the newly defined *sisHtml* command against millions of archived web resources that reside on the Hadoop distributed file system (HDFS) and handle the streaming between the application processes and the HDFS. ToMaR supports an extensible set of implementations for different transport protocols like file http, file, hdfs. Tools that are provided as Debian packages by the SCAPE project come with a tool specification document by default[6].

[5]available at https://github.com/openplanets/scape-toolspecs/blob/master/toolspec.xsd

[6]available at https://github.com/openplanets/scape-toolspecs

C. Generating and Handling Input

The operations defined in the toolspec documents specify atomic operations that can be carried out at scale using ToMaR. A design goal in this context was to enable users to execute an operation on the cluster in a very similar way as it would be done on a desktop computer. ToMaR takes a plain text file as input specifying how one or multiple joined toolspec operations are applied to the payload data. As per the MapReduce model, the generated input file is broken into splits at execution time and processed independently by distributed mapper processes. ToMaR's input file format can be split between lines, and each line of a split (called a record) is then processed by a map task. When generating an input file, there are a number of options one should consider in order to achieve good performance on a cluster, as shown in figure 3:

- Toolspec inputs: The ToMaR input file is used to specify the sequence of commands to be executed by a ToMaR job. It also selects the way IO should be handled and specifies concrete values for file placeholders defined by the tool specification documents. Many applications, however, implement custom workflows allowing one to process multiple files recursively or in batch mode minimizing application startup overheads. Exploiting tool-specific optimization strategies provides an important aspect that can be dramatically improve performance. An example for recursive processing is the File Information Tool Set (FITS) that has being utilized in the experiment described later in this paper.
- Toolspec output: The output generated by a record that is processed by a map task is written to standard output and/or files on the HDFS, as defined by the corresponding tool specification document. ToMaR can, however, be extended to pass the output values to one or multiple reducers. This can be very helpful to aggregate the produced results and/or to store data within a data base or container file. Current experiments deal with directly writing output into HBase and Hadoop SequenceFiles from ToMaR without requiring an additional MapReduce job to be run.
- Split size: ToMaR is configurable with respect to the number of mappers that are used to process the input file. This provides an important parameter as it determines the size of the input file splits that are shipped to the map tasks. For very short running operations less map tasks (processing n records) should be used than for long running operations, even on large clusters. Optimizing the tradeoff between number of mappers and split size is however subject to individual application fine-tuning, as shown in section IV.
- Chained operations: Another method to enhance the performance of a ToMaR based workflow is to combine multiple tool invocations per record using IO pipes,

```
#one file per record
fits filexml --input="hdfs:///1.tif" --output="hdfs:///1.tif.xml"
fits filexml --input="hdfs:///2.tif" --output="hdfs:///2.tif.xml"

#multiple files per record
fits dirxml --input="hdfs:///./" \
  --inputlist="hdfs:///1.tif,hdfs:///2.tif" \
  --output="hdfs:///./" \
  --outputlist="hdfs:///1.tif.fits.xml,hdfs:///2.tif.fits.xml"

#IO streaming
"hdfs:///1.tif" > mbx extract > "hdfs:///1.tif.cvs"

#multiple operations per record
"hdfs:///1.tif" > mbx extract | mbx filtr > "hdfs:///1.tif.f.cvs"
```

Figure 3. Examples of possible ToMaR input files: (a) processes one file or directory per record, (b) processes n files per record by utilizing recurse processing supported by the tool, (c) passes IO between the process and HDFS by redirecting input and output streams, (d) combines toolspec operations using IO pipes.

which is supported by ToMaR. Chained operations allow us to reduce the MapReduce overhead which occurs when multiple jobs have to be started in order to execute a workflow.

D. Comparing Related Approaches

Implementing custom MapReduce application provides a powerful way to create robust and scalable applications that are IO bound. Frameworks like PIG and HIVE [12] provide higher-level abstractions that enable users to analyze data that is organized in a data warehouse. For many domains, it is however not possible to implement the entire data acquisition and analysis workflow based on Java, MapReduce, and database abstractions. Porting many existing image processing tools to native Hadoop applications would not be feasible in many cases. As data-intensive computing frameworks provide the only realistic way to set-up large-scale archives and repositories, it is required to develop approaches that support legacy application integration. ToMaR has been developed to provide efficient legacy application integration for large-scale content and has been used to help analyzing content from major content providers. Other generic approaches like the Hadoop Streaming API have been very helpful to implement different steps of data analysis workflows but could not be used for legacy code integration. The Hadoop Streaming API, for example, supports the integration of scripts, IO pipes, Unix filters, and the automated generation of MapReduce jobs but lacks the support for legacy application integration. The Hadoop framework is designed to process input based on key/value pairs by automatically interpreting and splitting the input data. While this works perfect for processing text, it makes it difficult to process binary data. The Hadoop steaming API uses streams to read/write from/to HDFS but cannot be used with native applications that do not support IO streaming or HDFS file pointers. ToMaR is designed to support a number of different protocols for integrating legacy applications. It is obvious that optimized MapReduce applications can achieve a higher throughput than a generic wrapper that must handle the coordination of decoupled OS processes. However, by leveraging the Hadoop execution environment, ToMaR has proven to be scalable, robust, and with good performance for implementing many real world application scenarios where payloads are simply too big to be computed with existing applications on a single machines.

IV. User Case: Web Archive Analytics

A. Problem Statement

The Austrian National Library (ONB) collects the webpages within the domain .at, pages that are geographically sited in Austria, and also pages that have a specific connection with Austria. The ONB Web archive[7] presently amount to 32TB of data containing about 1.3×10^9 resources. Web archiving software like for example the Internet Archive's open-source Heritrix platform [13] can perform crawls in a distributed environment, considering different crawling strategies (focused, broad, experimental) and crawling policies. The crawled data is written to web archive container formats like the ARC or WARC File Format (ISO 28500) which hold the harvested data.

An interest of Web archives is to analyze trends in the size and content of the Web. The term *web content characterization* refers to the extraction of properties from web archives. A number of characterization tools exist that are commonly used in the digital library and archiving domain. However, only a sub-set of those tools are available as self-contained Java API, as for example the Apache Tika library[8]. The File Information Tool Set (FITS)[9] identifies, validates, and extracts metadata for various file formats. FITS wraps several third-party open source tools and produces normalized and consolidated output in the form of XML files. Although written in Java, FITS creates dependencies on a variety of libraries and file paths when being installed, as it works as an umbrella for a number of underlying software packages. Here, we report results from an experiment carried out by the ONB that applies FITS on sample data from the ONB Web archive. FITS, a tool kit that has been selected by the ONB to evaluate ToMaR, is of great interest for many people working in the archiving domain but difficult to handle at scale.

B. Workflow Development

For the presented experiment, ToMaR is used in concert with several software packages in order to create a profile of web archive content. The general idea of the workflow is to extract the content of ARC containers residing on HDFS. Per container, tens to hundreds of thousands of flat files with generic file names are generated. FITS is used to

[7]web site: http://www.onb.ac.at/ev/about/webarchive.htm
[8]available at: https://tika.apache.org
[9]available at: http://code.google.com/p/fits/

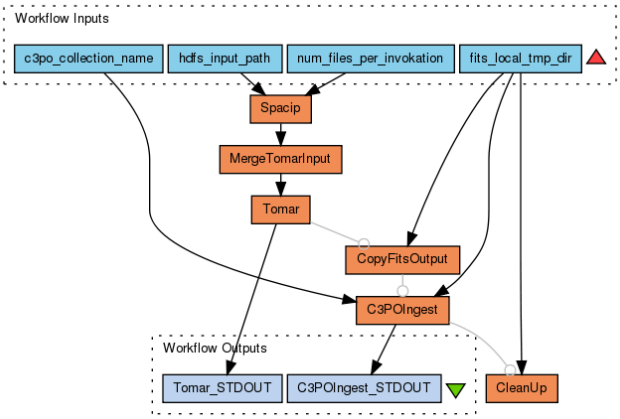

Figure 4. Web Archive FITS Characterization using ToMaR, available on myExperiment: www.myexperiment.org/workflows/3933.

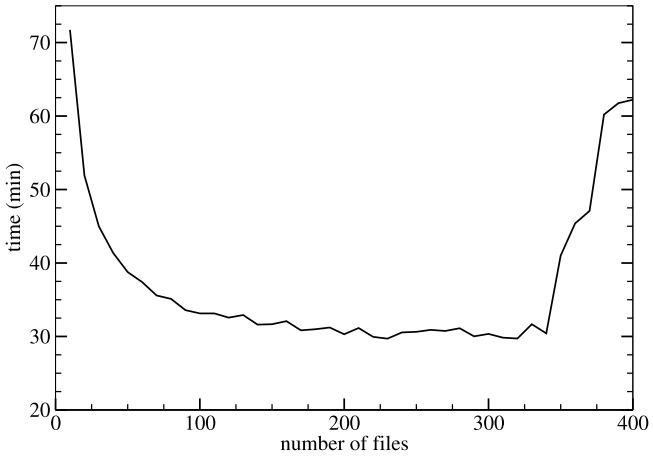

Figure 5. Execution times for a varying number of files in a record, which are handled by a single map invocation.

generate a metadata file for each resource that is used for subsequent content profiling. The extracted resources can be deleted from HDFS once the metadata files have been generated. For the experiment, Taverna has been used as the workflow management system [14] and used to submit and orchestrate jobs to the ONB development cluster. The workflow in figure 4 uses several SCAPE outcomes to create a profile of web archive content. It shows the complete process from unpacking a web archive container file to viewing aggregated statistics about the individual files it contains using the profiling tool C3P0[10]. The workflow uses the Hadoop application Spacip [11] to unpackage the ARC container files into HDFS and to create the input file which is subsequently used by ToMaR.

The inputs of this workflow are defined as follows:

- *c3po_collection_name*: The name of the C3P0 collection to be created.
- *hdfs_input_path*: A Hadoop Distributed File System (HDFS) path to a directory which contains text file(s) with absolute HDFS paths to ARC files.
- *num_files_per_invocation*: Number of items to be processed per FITS invocation.
- *fits_local_tmp_dir*: Local directory where the FITS output XML files will be stored

FITS comes with a command line interface API that allows a single file to be used as input to produce the FITS XML characterization result. However, if started individually for each file in the web archive, the start-up time of FITS including its sub-processes would result in a poor performance which is not applicable to the problem size. For the experiment, FITS was instead used in recursive mode using ToMaR's support for multiple input files per operation, which improved the performance dramatically.

[10]available at: https://github.com/peshkira/c3po
[11]available at: https://github.com/shsdev/spacip

C. Tuning the Input File

The question how many files should be processed per FITS invocation was addressed by an experiment. The workflow presented above has been embedded in a new workflow in order to generate a test series. A list of 40 values, ranging from 10 to 400 in steps of 10 files to be processed per invocation is provided as input using the *num_files_per_invocation* parameter. Taverna was used to iterate over the list of input values by combining the input values as a cross product and launching 40 workflow runs for the embedded workflow. 5 ARC container files with a total size of 481 Megabytes and 42223 individual files were used as input for this experiment. The 40 workflow cycles were completed in around 24 hours and led to the result shown in figure 5. The evolution of the execution time of the average and worst performing tasks per record shows linear characteristics, as illustrated in figure 6.

D. Performance on the Cluster

The cluster used in this experiment has one controller machine (Master) and 5 worker machines (Slaves). The master node has two quad-core CPUs (8 physical/16 HyperThreading cores) with a clock rate of 2.40GHz and 24 Gigabyte RAM. The slave nodes have one quad-core CPUs (4 physical/8 HyperThreading cores) with a clock rate of 2.53GHz and 16 Gigabyte RAM. Regarding the Hadoop configuration, five processor cores of each machine have been assigned to Map Tasks, two cores to Reduce tasks, and one core is reserved for the operating system. This is a total of 25 processing cores for Map tasks and 10 cores for Reduce tasks. The best result achieved on the cluster was an execution time of about 30 minutes (1782 seconds), while the execution on a single worker node took about 9 hours (32148 seconds). As shown in figure 5, ToMaR's overall execution time automatically stabilizes at an execution time of about 30 minutes for a record size between 150 to 350

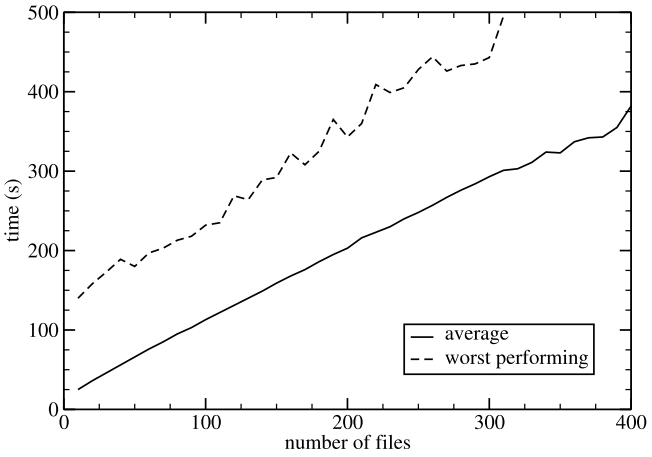

Figure 6. Evolution of execution time per operation for a varying amount of processed files.

web resources. Overheads introduced by application are negligible in this area.

V. Conclusion

Hadoop provides scalability, reliability, and robustness supporting the processing of very large volumes of unstructured data that does not fit on a single machine. Data and applications must however be made compliant with the execution environment and programming model. Our intention was to provide a generic MapReduce application allowing content providers like libraries and archives to execute legacy applications on a Hadoop cluster in a similar way like done in a desktop environment. This functionality is in particular required to generate large volumes of metadata from binary content which can then be imported into an archival information system or a data warehouse using well known data-intensive computing techniques. ToMaR is being evaluated by major institutions to support the scalable processing of large volumes of content including high-resolution scans of digitized books, millions of web archive resources, as well as with scientific data from large-scale facilities in the context of the EC project SCAPE. The major design goal of ToMaR is firstly to provide flexibility so that various applications coming from different content domains can be integrated in a way that they run efficiently on a cluster, and secondly to generate data that is integrated and directly usable with data-intensive computing frameworks like provided by the Apache Hadoop eco-system.

Acknowledgment

Work presented in this paper is primarily supported by European Community's Seventh Framework Programme through the project SCAPE under grant agreements No 270137.

References

[1] "Reference Model for an Open Archival Information System (OAIS) : Recommendation for Space Data System Standards : CCSDS 650.0-B-1," Tech. Rep., Jan. 2002. [Online]. Available: http://public.ccsds.org/publications/archive/650x0b1.pdf

[2] P. Van Garderen, "Archivematica: Using micro-services and open-source software to deliver a comprehensive digital curation solution," *Proc. of the 7th International Conference on Preservation of Digital Objects (iPres 2010)*, 2010.

[3] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Commun. ACM*, vol. 51, pp. 107–113, January 2008.

[4] K.-H. Lee, Y.-J. Lee, H. Choi, Y. D. Chung, and B. Moon, "Parallel data processing with mapreduce: A survey," *SIGMOD Rec.*, vol. 40, no. 4, pp. 11–20, Jan. 2012. [Online]. Available: http://doi.acm.org/10.1145/2094114.2094118

[5] A. F. Gates, O. Natkovich, S. Chopra, P. Kamath, S. M. Narayanamurthy, C. Olston, B. Reed, S. Srinivasan, and U. Srivastava, "Building a high-level dataflow system on top of map-reduce: The pig experience," *Proc. VLDB Endow.*, vol. 2, no. 2, pp. 1414–1425, Aug. 2009.

[6] Mahout: Scalable machine-learning and data-mining library, http://mapout.apache.org.

[7] R. Schmidt and M. Rella, "An approach for processing large and non-uniform media objects on mapreduce-based clusters," in *Proceedings of the 13th international conference on Asia-pacific digital libraries*, ser. ICADL'11. Springer-Verlag, 2011, pp. 172–181.

[8] First Alpha Release of Fedora 4 from Fedora Futures, https://github.com/futures/fcrepo4/releases/fcrepo-4.0.0-alpha-1.

[9] Apache Hadoop Project, http://hadoop.apache.org/.

[10] R. Schmidt, "An architectural overview of the scape preservation platform," in *Proc. of the Ninth Int. Conf. on Preservation of Digital Objects (iPres12)*, Toronto, Canada, October 2012.

[11] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The eucalyptus open-source cloud-computing system," in *Cluster Computing and the Grid, 2009. CCGRID '09. 9th IEEE/ACM International Symposium on*, May 2009, pp. 124–131.

[12] A. Thusoo, J. Sarma, N. Jain, Z. Shao, P. Chakka, N. Zhang, S. Antony, H. Liu, and R. Murthy, "Hive - a petabyte scale data warehouse using hadoop," in *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*, 2010, pp. 996 –1005.

[13] G. Mohr, M. Stack, I. Rnitovic, D. Avery, and M. Kimpton, "Introduction to heritrix," in *4th International Web Archiving Workshop*, 2004.

[14] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat *et al.*, "Taverna: a tool for the composition and enactment of bioinformatics workflows," *Bioinformatics*, vol. 20, no. 17, pp. 3045–3054, 2004.