Distributed OAIS-Based Digital Preservation System with HDFS Technology

Nikita Voinov, Pavel Drobintsev, Vsevolod Kotlyarov
Peter the Great Saint-Petersburg Polytechnic University
Saint-Petersburg, Russia
{voinov, drob, vpk}@ics2.ecd.spbstu.ru

Igor Nikiforov
Dell EMC/Peter the Great Saint-Petersburg Polytechnic University
Saint-Petersburg, Russia
igor.nikiforovv@gmail.com

***Abstract*—The paper describes architecture of a distributed OAIS-based digital preservation system which uses HDFS as a file storage system and supports wide distribution on a number of cluster's nodes. It is based on Apache Hadoop framework - a reliable open source solution with well horizontally scalable distributed architecture. Novelty of the proposed system is defined by the fact that none of existing OAIS digital preservation systems use HDFS storage for both structured and unstructured data archiving. Implementation of the system's prototype and results of its testing are also shown.**

I. Introduction

Providing secure storage for big data is extremely important challenge in the modern world as amount of generated data is constantly growing. World data volume is estimated in more than 8 zettabytes and will be doubled every two year [1]. When speaking about data volume it is also critical to mention that not only structured data exists and shall be preserved, but also unstructured content, which is about 80% of all existing data.

To solve issues with big data many companies rely on traditional RDBMS (relational database management system) [2] solutions, which show their excellence on data volume over terabytes. But when data exceeds petabytes, traditional storages start losing in performance and become unreliable as they are not scalable enough. Data size issues can be solved by NoSQL (Non Structured Query Language) [3] databases, however this technology does not provide enough flexible and powerful data management workflow in compare to the approach described below in the paper. For these reasons RDBMS and NoSQL implementations are not reviewed and out of scope of this article.

An alternative solution to satisfy business needs in structured and unstructured data secure storage is usage of digital preservation system. This is a system for storage of digital documents which provides security, confidentiality, different access levels, revision history tracing, fast and easy search engine. There is an international standard and a reference model for digital preservation systems called OAIS (Open Archival Information System) [4]. Its goal is to formalize requirements for digital preservation systems. It describes standard architecture and functionality of a digital archive. Many famous digital preservation projects are based on this model. Among them are RODA [5], [6] created in cooperation with the Portuguese National Archives; Archivematica [7] used by The National Archives of the Czech Republic; DRI (the Digital Repository of Ireland) [8]; SPAR in the National Library of France [9]; InfoArchive [10] initially developed by Dell EMC company and recently acquired by OpenText company.

The analysis of existing solutions revealed that InfoArchive is the most powerful, scalable and secure storage, however it is a proprietary software. The rest vendors are small fishes. Although some of them provide open source solutions, technologies used for their digital storage implementations put significant constraints on the level of horizontal scaling in terms of supported amount of distributed nodes in a storage. This hampers effective storage distribution on high-performance clusters and supercomputers.

In order to provide scalable, reliable, secure, open source solution which also satisfies to the OAIS standard, this work proposes usage of Apache Hadoop [11] technology with distributed HDFS (Hadoop Distributed File System) [11] file system, which is not applied in any existing solutions observed (Table I).

TABLE I. OAIS-BASED SOLUTIONS FOR DIGITAL PRESERVATION

	RODA	**Archivematica**	**DRI**
Proprietarity	Open-source	Open-source	Proprietary
Programming Language	Java	Python	Java
Ongoing Project	No	No	Yes
Data Storage	Fedora Commons or Native file system	Native file system	CephFS
Search Engine	No	No	No
Data Analytics	No	Yes	No
Horizontal Scaling	Limited	Limited	Yes
Message Brokers	No	No	No

HDFS provides:

- distributed cluster architecture;
- high reliability at installation;
- horizontal scaling to up to thousands of nodes;
- safe data storage;
- multiple interfaces for interaction;

- multilevel API;
- dynamic configuration.

Presently HDFS is likely to be the best solution for secure and distributed data storage for both structured and unstructured content [12-14].

The goal of this work is to present an implementation of a digital preservation system prototype based on the OAIS model and HDFS storage for full support of horizontal scaling, fast search, secure data receipt and storage.

II. Open Archival Information System

The OAIS is an international standard of digital preservation. It describes standard architecture and functionality of a digital archive. On different stages of data lifecycle the OAIS introduces the following artifacts: data submitted into the archive (SIP - Submission Information Package [4]), archived data (AIP - Archival Information Package [4]), disseminated data (DIP - Dissemination Information Package [4]).

Functional model of the OAIS is represented in Fig. 1:

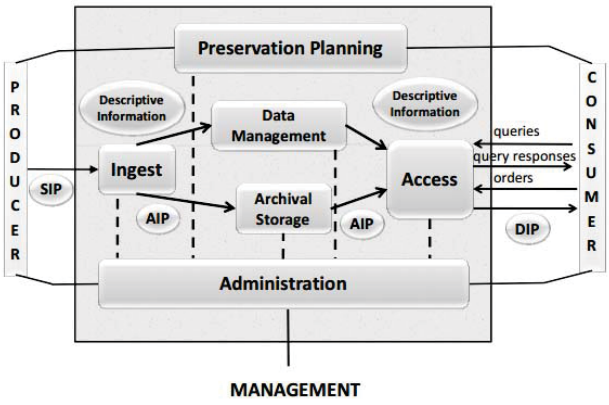

Fig. 1. OAIS functional entities [4]

The roles listed below are considered to be external regarding to the OAIS:

- Producer - sources of the archive gathering and their informational systems which produce data and transfer data for preservation;
- Consumer - people or systems interacting with archive services to find and obtain archived data; according to the standard there is a specific user group called ”designated community” - a group of users who shall be able to understand archived data;
- Management of the organization (host of the archive) who defines the archive strategy, aims and tasks.

Described below are the main modules of the OAIS internal architecture.

A. The Ingest functional entity

This entity provides the services and functions to accept SIPs from Producers (or from internal elements under Administration control) and prepare the contents for storage and management within the archive [4]. Ingest functions include:

- receiving SIPs;
- performing quality assurance on SIPs;
- generating an AIP which complies with the archive’s data formatting and documentation standards;
- extracting descriptive information from the AIPs for inclusion in the archive database;
- coordinating updates to Archival Storage and Data Management.

B. The Archival Storage functional entity

Provides the services and functions for the storage, maintenance and retrieval of AIPs [4]. Archival Storage functions include:

- receiving AIPs from Ingest and adding them to permanent storage;
- managing the storage hierarchy;
- refreshing the media on which archive holdings are stored;
- performing routine and special error checking;
- providing disaster recovery capabilities, and providing AIPs to Access to fulfill orders.

C. The Data Management functional entity

It provides the services and functions for populating, maintaining, and accessing both descriptive information which identifies and documents archive holdings and administrative data used to manage the archive [4]. Data Management functions include:

- administering the archive database functions (maintaining schema and view definitions, and referential integrity);
- performing database updates (loading new descriptive information or archive administrative data);
- performing queries on the data management data to generate query responses and producing reports from these query responses.

D. The Administration functional entity

Provides the services and functions for the overall operation of the archive system [4]. Administration functions include:

- soliciting and negotiating submission agreements with Producers;
- auditing submissions to ensure that they meet archive standards;
- maintaining configuration management of system hardware and software.

It also provides system engineering functions to monitor and improve archive operations, and to inventory, report on, and migrate/update the contents of the archive. It is also responsible for establishing and maintaining archive standards and policies, providing customer support, and activating stored requests [4].

E. The Preservation Planning functional entity

This entity provides the services and functions for monitoring the environment of the OAIS, providing recommendations and preservation plans to ensure that the information stored in the OAIS remains accessible to, and understandable by, the designated community over the long term, even if the original computing environment becomes obsolete [4]. Preservation Planning functions include:

- evaluating the contents of the archive and periodically recommending archival information updates;
- recommending the migration of current archive holding;
- developing recommendations for archive standards and policies;
- providing periodic risk analysis reports;
- monitoring changes in the technology environment and in the designated community's service requirements and knowledge base.

Preservation Planning also designs information package templates and provides design assistance and review to specialize these templates into SIPs and AIPs for specific submissions. Preservation Planning also develops detailed migration plans, software prototypes and test plans to enable implementation of Administration migration goals [4].

F. The Access functional entity

It provides the services and functions that support Consumers in determining the existence, description, location and availability of information stored in the OAIS, and allowing Consumers to request and receive information products [4]. Access functions include:

- communicating with Consumers to receive requests;
- applying controls to limit access to specially protected information;
- coordinating the execution of requests to successful completion;
- generating responses (DIP, query responses, reports) and delivering the responses to Consumers.

III. Conceptual approach to the architecture of the digital preservation system

It is supposed in this work that the main feature of the digital preservation system being developed is that it shall be distributed. This means that it shall support simultaneous execution on specified number of cluster's nodes. Interaction and synchronization between the nodes shall be supported as well. The system shall support horizontal scaling to handle extra storage equipment (hardware-software unit which can be connected to the cluster) when there is no free space left in the data storage. In case of increased server load because of high intensity of input data flow, a new server can be added to distribute load between different nodes and make overall working process more balanced.

Horizontal scaling in the system is provided by automated configuration of distributed storage with configuration files.

The prototype supports secure storage of stored data. Stored packages have several access levels. A user with insufficient rights cannot get access to the data. Data access rights are implemented by Kerberos authentication system [15].

Data replication is proposed to increase fault tolerance. By that the system keeps operating even if some node is damaged. Data replication is performed on the storage level and the system can configure replication coefficient which specifies number of nodes to locate each unit. By default replication coefficient supported by HDFS equals 3.

Errors and data loss shall be eliminated during data transmission. Message brokers are proposed to do that. They provide required level of security and integrity during data transmission. Packages received at the preservation system input pass through message broker. Thus, horizontal scaling with increased input data flow is achieved in general system's configuration.

The system shall also provide easy search among stored data. A search module is used in the prototype. A data package is indexed by the search engine after its validation in the archive module. Only structured text information from the package is being indexed. Search engine and stored indexes are distributed in the cluster as a cloud. Usage of search library and back index approach provide fast search through stored data. Search index shall be stored in distributed cluster system to keep support for good horizontal scaling.

Software components of the system are the core (the archive module), connectors and the client part (Fig. 2). Connector downloads data from some data source, creates packages from this data and passes them to the system's core entry. For testing purposes connectors obtained data from Twitter and Facebook social networks.

The core (the archive module) is responsible for package receipt, its indexing and writing into the storage (Fig. 2). The core also contains logging module and module for managing stored data.

Client interface is used to interact with data received by the system (Fig. 2). Searching servlet allows to input key words into special form and search for them through structured indexed data. Web interfaces are also supported to debug particular system components or download specific packages.

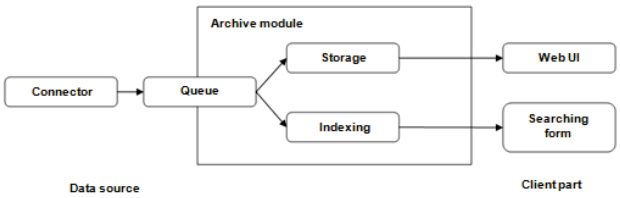

Fig. 2. Main components of the preservation system

The cores and connectors can be run on any node using script (Fig. 3).

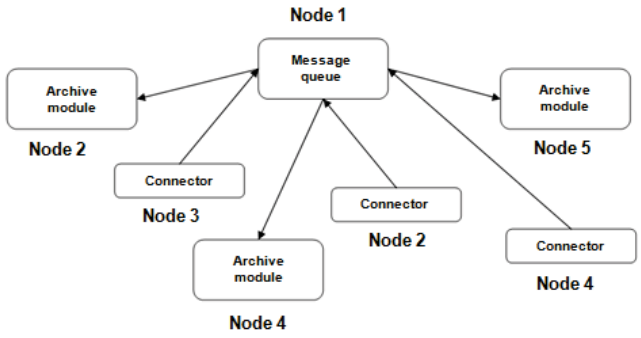

Fig. 3. Distributed configuration of the preservation system

A. The Archive module

The main part of the system is the archive module. Its main task is to download data packages into the preservation system. Its lifecycle can be divided into 3 steps: writing into the system; storing and administrating in the system; reading from the system.

Writing is performed as following. SIP constructed in accordance with the OAIS standard is sent to the archive module through message broker. Then this package moves into the unit where its construction is checked for correctness. In case of failed check the packed is rejected without writing in the storage and indexing by search engine, otherwise it is written and indexed.

Storing and administrating is performed during system execution. The archive module contains logging unit, reports generation unit and a unit for managing data in the storage. Logging is required to trace system state and possible errors during execution. Reports generation provides convenient interface for providing compact information about all data in the storage. Managing data in the storage provides storing and removal specific packages depending on metadata contained inside them.

Reading from the system is possible whether by searching through structured information in the packages using searching servlet or by extracting whole packages from the system using special web UI interface.

B. Data storage file system

To provide high quality infrastructure of data storage modern cloud storages shall provide the following features:

- high reliability;
- constant data availability;
- fast access with minimal delays;
- low cost;
- additional features: clone, snapshots, etc.

Neither RAID storages nor data storage systems can provide all these features simultaneously. As a result, software-defined storage is getting a very popular approach in data storage industry.

High reliability and calculation speed are the most important criteria when selecting a file system for a preservation system. This can be achieved by replication, i.e. multiple copies of data blocks and their distribution among cluster's nodes. Data load shall be distributed among several machines. Files in the file system shall be written only once and arbitrary modifications shall not be allowed. To read and write files a specific interface for interaction with file system shall be used. Data security support shall also be provided to avoid unauthorized access to confidential information.

C. Search engine

Automatic indexing of structured data and interface for convenient search among indexed data are required for the preservation system. Search engine shall also provide the following features:

- full text search;
- highlighting search results;
- integration with data bases;
- language for requests with support of structured search on the same level as text search;
- support of multiple formats over HTTP including JSON, XML, CSV and libraries for other programming languages;
- web-interface for administration;
- replication to increase request processing speed;
- search through data distributed over multiple nodes;
- dynamic clustering of search results;
- caching requests, filters and documents.

D. Message broker

Message broker is typically a set of applications each of which performs processing of a particular step within message exchange: receive a message, put a message on a queue and transfer a message to the process responsible for execution.

Message broker is an intermediate level between different services. It significantly reduces the time on a message transfer as the time consuming tasks are distributed among working processes intended exactly for these particular tasks. Message broker provides secure channel for message transfer between applications.

Message broker in preservation system shall guarantee stability and scaling of the queue as it is the key node of the

whole system. It shall also be open source and implement AMQP protocol [16] which provides libraries for all main programming languages and platforms. Also required are high speed of service, security of transferred data and support of horizontal scaling for cluster architecture. Message broker shall effectively distribute data load among clients in automatic mode as well.

IV. Implemented Prototype of the Preservation System

The prototype was implemented on a cluster with 6 nodes, where each node is a virtual machine (Fig. 4). VMware vSphere [17] infrastructure was used for this architecture. Each node runs Ubuntu 14.04 operating system.

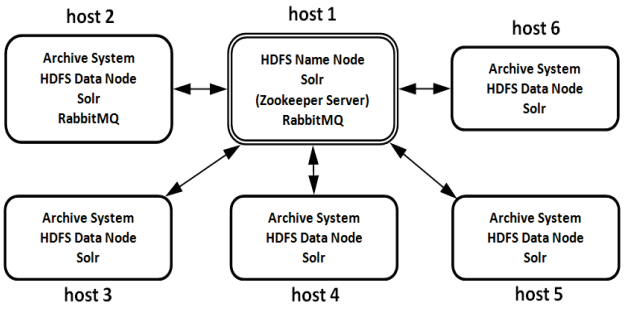

Fig. 4. Software distribution among cluster's nodes

The cluster contains installation of Apache Hadoop distributive from Cloudera company – CDH. The first node is the NameNode, other 5 nodes are DataNode.

RabbitMQ [18] is used in the system as a message broker to transfer generated SIPs to the archive module. It is installed on the first two cluster's nodes. RabbitMQ implements data exchange between system's components. Horizontal scaling is also supported for cluster architecture.

Apache Solr [19] is used as a search engine through archive packages as it is one of the most popular search toolsets. Solr provides distributed search and replication and supports scaling. Solr is deployed on all 6 nodes. Communication between the nodes is provided by Apache Zookeeper [20] server which is installed on the first cluster's node.

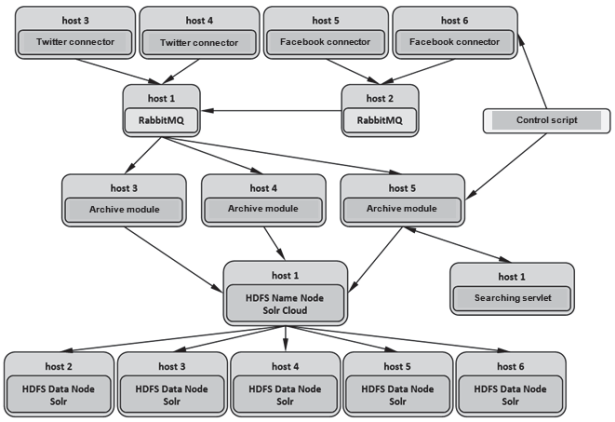

Fig. 5. Interaction between modules of the preservation system

Control script starts and stops the connectors and the cores of the preservation system in the cloud (Fig. 5). When connectors start execution, data packages generated by them are transferred to the message broker located on the nodes one and two. The system receives these packages from queues, loads them into the storage and indexes their contents.

A. Implementation of the archive module

Prior to data load in the archive module data packages shall be generated. For this purpose two connectors to Facebook and Twitter social networks were implemented (Fig. 6). The main task of the connectors is downloading data both structured and unstructured and preparing SIPs of specified format and size based on this data.

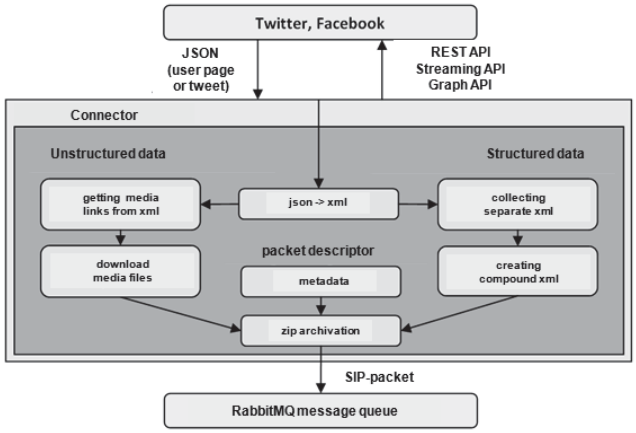

Fig. 6. Scheme of the connector

SIPs appear in the system through message broker. Correctness of the input data is checked in Ingest module (Fig. 7). Then a package is written into HDFS or rejected in case of some errors. All structured information passing through the archive module is indexed by search engine.

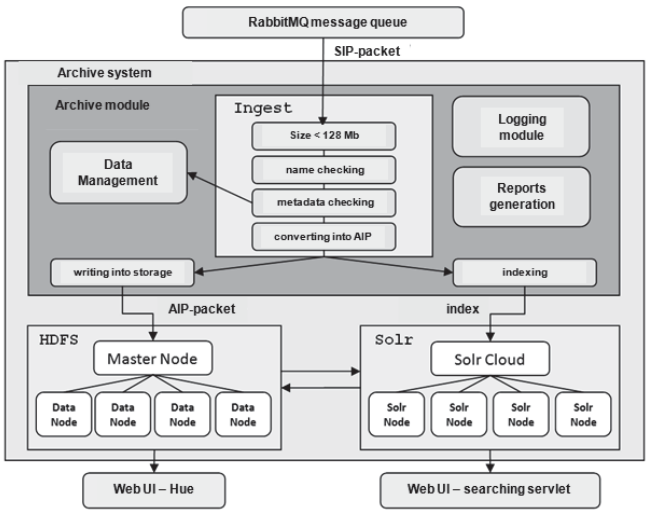

Fig. 7. Scheme of the archive module

Logging and Data Management modules are also involved in the system execution. Also supported is generation of reports based on information within the system.

Client part is represented as Web UI interfaces. Hue allows viewing the storage contents. While searching servlet performs searching through the indexed packages contained in HDFS. It is also possible to view the contents of a particular file.

B. HDFS storage

HDFS is a file system intended to store files of large size which are distributed by blocks among cluster's nodes. All blocks in HDFS (except the last block of a file) have one size and each block can be located on several nodes. The block size and its replication coefficient are defined in file settings. Replication ensures fault tolerance of the whole system to failures of separate nodes. Each file can be written into HDFS only once while input into the file can be performed by only single process at one time. Files organization within the name space is traditionally hierarchical: there is a root folder, subfolders and each folder can contain files and other folders.

Deploying instance of HDFS is supposed to have one name node, which stores metadata of a file system and information about blocks distribution, and a set of data nodes, which store all files blocks. Name node is responsible for processing operations on the level of files and folders: files opening and closing, folders manipulating. Data nodes process operations of data reading and writing. Name node and data nodes are supplied by web-servers to display current node state and to observe the contents of the file system. Administrating functions are available via command line interface.

C. Solr search library

Solr is an open source platform for full text search based on Apache Lucene project [21]. Except full text search it also provides highlighting search results, dynamic clustering, integration with data bases, processing documents of complex formats (Word, PDF). Solr is highly scalable due to support of distributed search and replication.

Solr is written in Java and launched as a separate web-application of full text search. It uses Lucene as a basis for searching and indexing. It contains HTTP/XML and JSON API which makes possible to use Solr with almost all popular programming languages. It also supports flexible setup and connection with external modules.

D. RabbitMQ as a messaging broker

RabbitMQ is open source message broker software (sometimes called message-oriented middleware) that implements the Advanced Message Queuing Protocol (AMQP). The RabbitMQ server is written in the Erlang programming language and is built on the Open Telecom Platform (OTP) framework for clustering and failover. Client libraries to interface with the broker are available for all major programming languages (Java, .NET, Perl, Python, Ruby, PHP, etc.) RabbitMQ is produced under Mozilla Public License. It supports horizontal scaling for deploying cluster architecture.

One of the main reasons to use RabbitMQ is its implementation over Erlang/OTP platform which guarantees stability and scaling of the queue as the key node of the whole system. Another reason is publicity of Mozilla Public License and AMQP with libraries available for all programming languages and platforms including Node.js

V. RESULTS

Developed prototype of the system has successfully passed all testing stages.

Module testing revealed that all program components are working properly: connectors can download data and create data packages while the archive module can write packages into the storage.

Integration testing was conducted to check connection between the modules. All components including message queue were run and their communication was observed. A set of data packages was successfully transferred from connectors to the archive module via message queue.

Functional testing checked miscellaneous use cases of the system execution. Components were run on different nodes, 100 data packages of 50 Mb each were generated and many search requests were run via searching servlet.

During load testing connectors and archive modules were run on different nodes and the time spent on single package writing into the storage was measured.

Consider a test case when 600 packages of 50 Mb each shall be generated and each cluster node runs one connector and one archive module. Results of this test case are shown in Table II.

TABLE II. SYSTEM LOAD DEPENDING ON NUMBER OF NODES IN USE

N_{nodes}	P_{node}	V_{node}	T_1	T_2	T_{sum}	U_{node}	U_{sum}
1	600	30000	244,86	5,69	250,55	119,74	119,74
2	300	15000	122,43	4,51	126,94	118,17	236,33
3	200	10000	81,62	3,83	85,45	117,03	351,08
4	150	7500	61,22	3,42	64,64	116,04	464,14
5	120	6000	48,97	2,78	51,75	115,94	579,69
6	100	5000	40,81	2,36	43,17	115,82	694,93

Where:

- N_{nodes} - number of cluster nodes in use;
- P_{node} - number of packages which shall be downloaded on one node; the load is distributed evenly and equals $P_{node}=600/N_{nodes}$;
- V_{node} (in Mb) - average size of data loaded on one node; calculated as $V_{node}= P_{node} * 50$;
- T_1 (in minutes) - overall time on loading data and preparing packages on one node;
- T_2 (in minutes) - overall time on verification, writing packages into file system and their indexing on one node;
- T_{sum} (in minutes) - overall time on constructing all packages on one node; $T_{sum} = T_1 + T_2$;

- U_{node} (in Mb/min) - archiving speed on one node; $U_{node} = V_{node} / T_{sum}$;
- U_{sum} (in Mb/min) - overall archiving speed on the whole cluster; $U_{sum} = U_{node} * N_{nodes}$.

The results show that increasing number of nodes leads to lower load of each node (considering constant number of packages) and the most productive configuration with the fastest generation and writing is 5 connectors and 2 archive modules.

VI. Conclusion

So, in the scope of this work the architecture of a digital preservation system was proposed, analyzed and used for creation of a prototype of the OAIS-based system with full support of horizontal scaling on distributed architecture due to HDFS storage.

The system architecture allows to have reliable distributed storage as well as powerful management capabilities provided by the OAIS standard.

Usage of HDFS as a storage for data allows not only to have robust store, but also perform and execute different analytical tasks and jobs which can be developed on customer side, without necessity to transfer the data out of customer location.

Having the system satisfy the OAIS standard makes sure that required data workflow is implemented and the data can be stored for tens of years.

The project was held in collaboration of Peter the Great Saint-Petersburg Polytechnic University with Dell EMC company. The aim and goals of the project were fully achieved and the project was considered to be successfully completed.

References

[1] X. Jin, B.W. Wah, X. Cheng and Y. Wang, “Significance and challenges of big data research”, *Big Data Research*, vol. 2, issue 2, June 2015, pp. 59-64.

[2] B. Rahnama, *Hierarchical data in RDBMS: a new horizon for data storage and retrieval*. LAP LAMBERT Academic Publishing, 2011.

[3] P.J. Sadalage and M. Fowler, *NoSQL distilled*. Addison-Wesley Professional, 2012.

[4] The Consultative Committee for Space Data Systems, Reference Model for an Open Archival Information System (OAIS), CCSDS 650.0-M-2, Magenta Book, June 2012, Web: https://public.ccsds.org/pubs/650x0m2.pdf.

[5] RODA Community, Web: http://www.roda-community.org.

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

[7] Archivematica: open-source digital preservation system, Web: https://www.archivematica.org.

[8] Digital Repository of Ireland, Web: http://dri.ie.

[9] Preservation of digital material: the SPAR project, Web: http://www.bnf.fr/en/preservation_spar_old/s.preservation_spar_realization_old.html?first_Art=non.

[10] OpenText company official website, Web: http://documentum.opentext.com/infoarchive.

[11] T. White, *Hadoop: the definitive guide, third edition*. O’Reilly, 2012.

[12] B.A. Jurik, A.A. Blekinge, R.B. Ferneke-Nielsen and P. Møldrup-Dalum, “Bridging the gap between real world repositories and scalable preservation environments”, *International Journal on Digital Libraries*, vol. 16, issue 3-4, May 2015, pp. 267-282.

[13] J. Lin, M. Gholami and J. Rao, “Infrastructure for supporting exploration and discovery in web archives”, *in Proc. of the 23rd International Conference on World Wide Web*, Apr. 2014, pp. 851-856.

[14] L. Medjkoune, S. Barton, F. Carpentier, J. Masanès and R. Pop, “Building scalable web archives”, *Archiving 2014 - Final Program and Proceedings*, May 2014, pp. 138-143.

[15] B. Spivey and J. Echeverria, *Hadoop security*. O'Reilly Media Inc., 2015.

[16] AMQP - Advanced Message Queuing Protocol, Web: http://www.amqp.org/.

[17] L. Dekens, J. Medd, G. Sizemore, B. Graf, A. Sullivan and M. Boren, *VMware vSphere powerCLI reference, 2nd edition - automating vSphere administration*. Wiley / Sybex, 2015.

[18] S. Boschi and G. Santomaggio, *RabbitMQ cookbook*. Packt Publishing, 2013.

[19] S. Mohan, *Apache Solr high performance*. Packt Publishing, 2014.

[20] F. Junqueira and B. Reed, *ZooKeeper - distributed process coordination*. O'Reilly Media, 2013.

[21] E. Ng and V. Mohan, *Lucene 4 cookbook*. Packt Publishing, 2015.