DELIVERABLE

Project Acronym:	**Europeana Cloud**
Grant Agreement number:	**325091**
Project Title:	**Europeana Cloud: Unlocking Europe's Research via The Cloud**

D2.4 - Prototype of Content Cloud

(incorporating D2.5 - Prototype of Metadata Cloud and Core Services)[1]

Authors:

Aleksandra Nowak (PSNC)
Pavel Kats (EF)
Marcin Werla (PSNC)
Alastair Dunning (TEL / EF - Review)
Carlo Menghini (ISTI - Review)

Revision: 1

[1] D2.4 and D2.5 have been incorporated into the same document - the technical design of Europeana Cloud no longer distinguishes between metadata and content, thus having separate documents did not make sense. A full explanation is given in the introduction

Project co-funded by the European Commission within the ICT Policy Support Programme		
Dissemination Level		
P	**Public**	**P**
C	**Confidential, only for members of the consortium and the Commission Services**	

Revision History

Version	*Status*	*Author*	*Partner*	*Date*	*Changes*
0.1	*First Version*	*Aleksandra Nowak* *Pavel Kats*	*PSNC* *EF*	*19/01/2015*	
0.2		*Aleksandra Nowak* *Marcin Werla*	*PSNC* *PSNC*	*22/01/2015*	*Deployment section improved*
0.3		*Pavel Kats*	*EF*	*Jan 2015*	*Further additions*
1.	*Publish Version*	*Pavel Kats* *Aleksandra Nowak* *Marcin Werla*	*EF* *PSNC* *PSNC*	*Jan 2015*	*Adopting reviewers' comments*

Statement of originality:

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

Abstract

This deliverable describes the prototype of the technical infrastructure of the Europeana Cloud project. The prototype implements the architectural design submitted in D2.2 Architectural Design Document. The reference installation, described here in detail, is hosted by PSNC. This is a distributed software system consisting of developed in-house and standard open-source components responsible for various aspects of functionality, outlined in the design document. Because scalability is the hallmark of cloud computing a special emphasis was put on setting up this installation to make it easily scaleable to varying levels of workload. While the scalability assumption is still to be tested, we measured the performance of the current setup and developed two pilot applications that performed migration from existing systems for storing cultural heritage data to the current installation of the infrastructure. In general, the measurements were inline with the expectations but raised some specific issues for further investigation. The hardware and software setup, the performance benchmark and the pilots are described in detail in the deliverable.

In 2015 the system will be put to the test by integration with production systems of the partners. We consider this deliverable to be the first candidate (alpha) for this integration.

Content

Introduction

This deliverable describes the progress of WP2 of the Europeana Cloud project, responsible for the development of the technical infrastructure. Our previous major deliverable, Architectural Design Document D2.2 described the architecture of the system. Since then, the alpha version of the infrastructure was developed and a production instance of this infrastructure was installed in PSNC. Below we describe principles which guided us in creating this instance, a suite of performance tests for the infrastructure and two pilot migration applications that were developed.

In 2015 Europeana Foundation, Poznan Supercomputing and Networking Center and The European Library will start integrating Europeana Cloud into their aggregation systems.

A remark concerning this deliverable, **D2.4 - Prototype of Content Cloud** and the next one, **D2.4 Prototype of Metadata Cloud and Core Services** is in place. Because of the way the system treats various types of data, it does not distinguish between metadata and content. For example, it would treat in the same a DC metadata record and a JPEG access copy of an image. Therefore, the distinction between content cloud and metadata cloud, initially planned in the project, and reflected in the deliverables, is not relevant anymore. Besides, the core services are an essential part of the system - cannot be delivered without them. Therefore, there is no real difference between **D2.4** and **D2.5** as planned in the DOW. We suggest to consider the current deliverable as these two altogether.

Summary

This deliverable is an interim report on development and deployment of the technical infrastructure of the Europeana Cloud project. We implemented the system according to the design and set up the first installation of production scale that is considered the alpha version suitable for integration with production systems of the clients. We ran a performance test on the system and started tested it *de facto* with two pilot applications migrating real data from existing aggregation infrastructures. Our conclusions are that the system is stable and performant to continue further tests and integration development. However, more tests should be run in the future to estimate its scalability and performance during real-time load of clients.

Europeana Cloud consists of distributed client-facing frontend services and backend systems. Performance analysis of the entire system should cover both types of components. The results of the our performance benchmark did not flag any significant performance issues with the systems. However, they made clear that some parts of the system, such as handling large files, still need some further investigation and optimization.

We were happy to notice that during most of the test time average CPU consumption of the application machine was under 26%. Memory consumption was stable as well. This indicates that there can be more services deployed on current application machines. Also the Apache Cassandra cluster worked efficiently during the entire test. Linear change in the number of threads resulted in a linear increase of the traffic managed by the cluster. This is a very promising result in terms of scalability of Europeana Cloud services.

Unfortunately we discovered bottlenecks while uploading large files. We noticed that the application machine reached the limit of 50 MB/s of outgoing traffic. We assume that it has to do with Openstack Swift limitations or configuration. The problem should be investigated further because it could lead to an bottleneck in the throughput of the entire system.

In the future another iteration of the benchmark should be planned. That time it should be run on multiple testing machines and probe multiple application machines with Europeana Cloud services deployed. That will give a good indication of the scalability of the entire system architecture.

Architecture

The architecture of the Europeana Cloud was designed following the process of gathering requirements which is described in detail in the previous technical deliverable D2.2. Figure 1 from that deliverable, shown here for convenience, shows the main parts of this architecture.

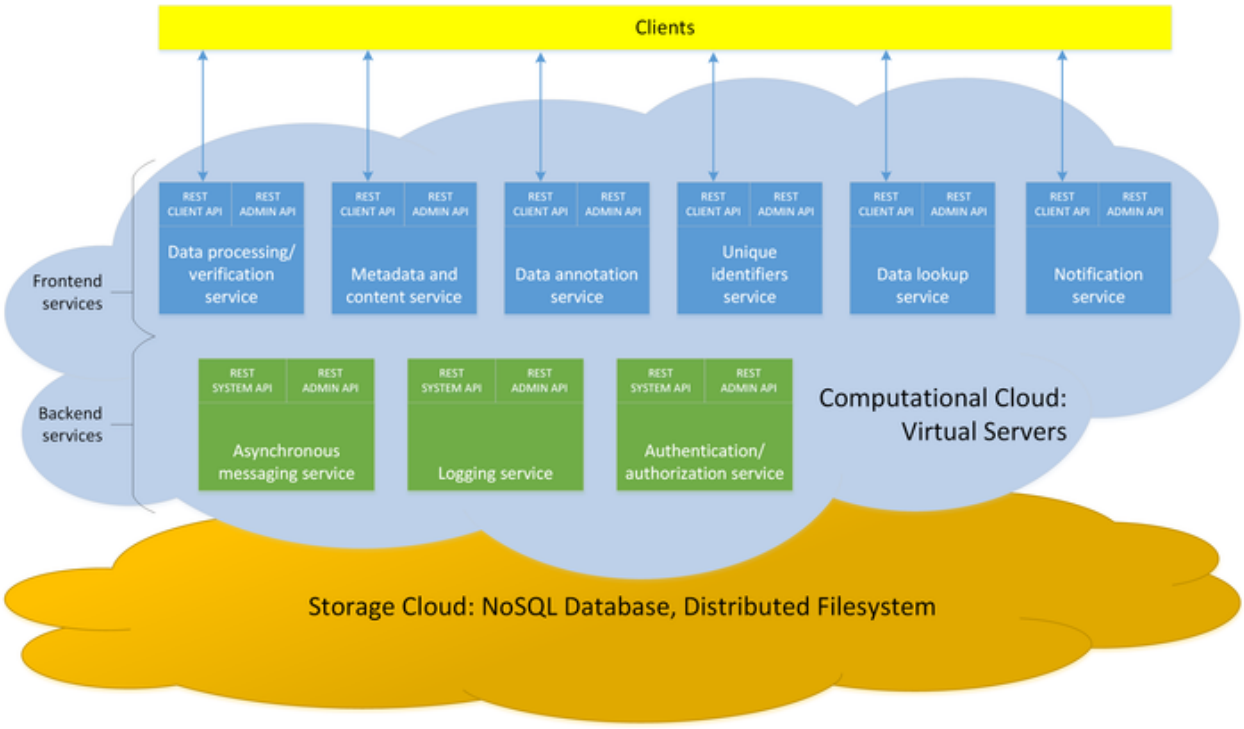

Figure 1: Europeana Cloud Service Architecture

To a client, the entire system appears as as a Software-as-a-Service (SaaS) system and can be used similarly to other SaaS services available on the Internet today.

A client can use several services, each one providing an API for its own functionality. Services are designed to be stateless in order to allow horizontal scalability. They also follow the standard REST service design approach.

On resources level, the system consists of two types of cloud:

- **Computational Cloud**
  to provide computing capacity for services executed by clients and other services
- **Storage Cloud**
  to provide storage capacity for services deployed in the Computational Cloud

The **Computational Cloud** hosts all the services which can be used by clients and therefore it is the one which is explicitly contacted by them (frontend services, see below). Also backend services are hosted there (e.g. logging service). The **Storage Cloud** operates behind the scenes on behalf of the clients because all the services use it.

The services in the **Computational Cloud** are arranged in two layers:

- **Frontend Services**
  which are directly available for the clients of the system. They are also called functional services (blue) because they cater specific functionality used by clients.
- **Backend Services**
  which are internal and are not available directly for end users, but are used by other services for administrative purposes. They are also called system services (green).

Deployment

The installation of Europeana Cloud at PSNC consists of several groups of servers acting as an integrated system, called clusters, on which services of the system or 3rd party software are installed. Efficient clustering is critical for building a successful distributed computing system. On the one hand, clusters are able to execute resource-intensive tasks, such as a heavy query or a batch update, quickly because all cluster´s machines work on the task altogether. On the other hand, the right balance between clusters should be created. Because resources are not unlimited, systems consisting of several clusters are to be built to avoid bottlenecks between clusters and skewed distribution of resources.

There are two types of machines used for the clusters:

1. Physical machines which provide the best of performance but do not allow for virtualisation (designated by rectangles)
2. Virtual machines with networked storage (designated by ellipses)

The infrastructure of Europeana Cloud consists of several independent components that act together to achieve high performance. We followed these principles to achieve a plausible result:

- The storage part of the system, the Storage Cloud, has to be performant as well as reliable. Therefore for it we used physical machines with a lot of of processing power and significant number of hard drives.
    - The database cluster can be easily scaled out using the out-of-the-box ability of the system to scale when a new node is added.

    - The storage cluster can not be scaled as easily because new nodes have to be configured. If scaling will be needed often, a standard scaling procedure will be prepared.
- The messaging system acts as intermediary between the services and will experience a lot of traffic. Thus it is designed an independent cluster built on physical machines. The messaging software has an built-in scaling mechanism so that this cluster can be enhanced when needed.
- The search index cluster consists of two machines, one of which is used for indexing new records and the other for searching. This in order to allow quick search operations while there is a massive indexing operation going on.
- Application services reside on their own cluster for quick identification of a performance bottleneck in one of them. Horizontal scaling mechanism should be designed to allow quick resolution of such bottlenecks, load balancing and failover. For the moment there is no natural clustering mechanism to group several physical services into one logical. It will be developed in the future and rely on Apache Zookeeper. This is on our list.

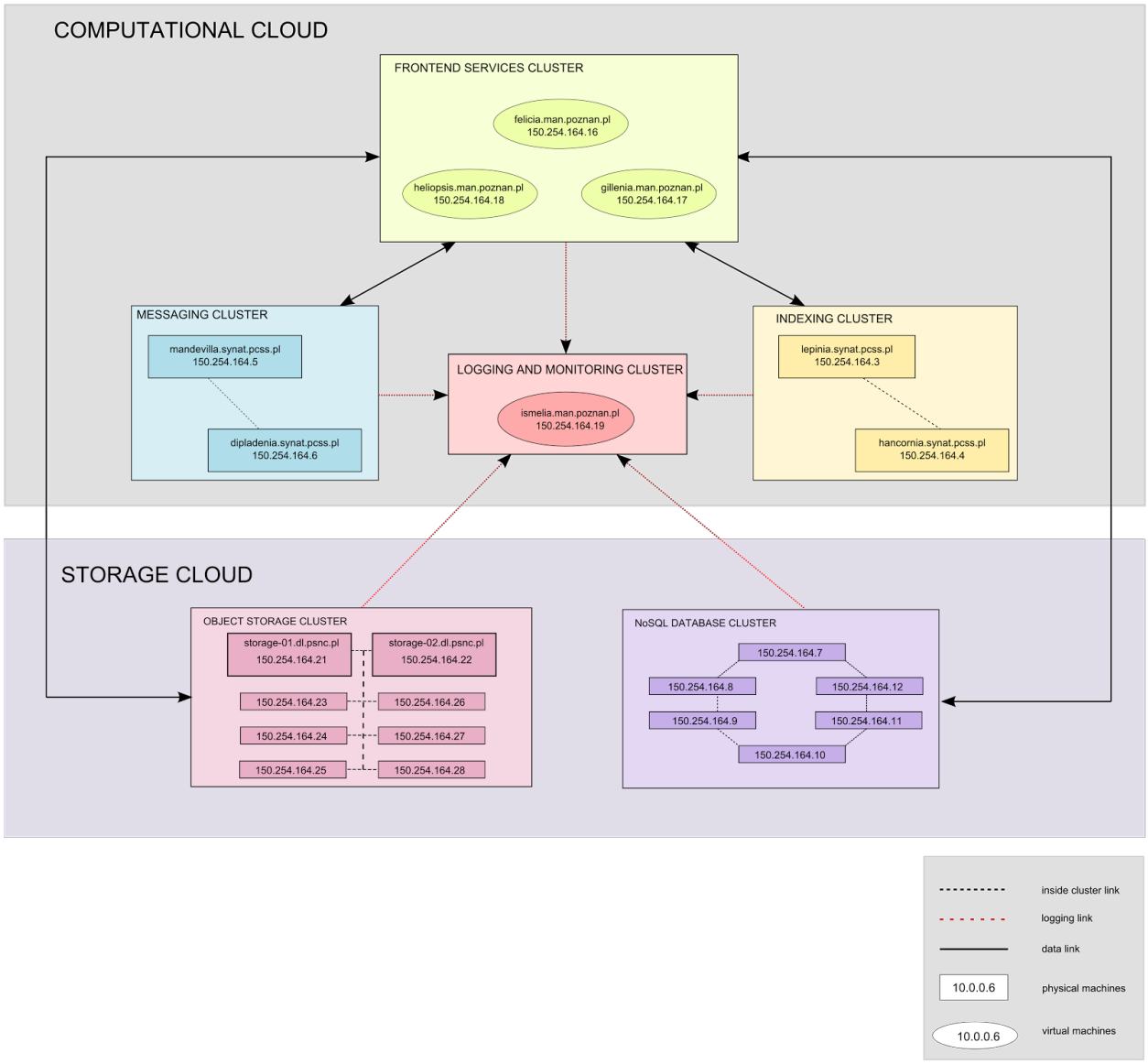

Figure 2: Europeana Cloud Deployment Scheme

Storage Cloud

Object Storage Cluster

Size	8 machines
Software	OpenStack Swift 1.10.0
Function	Storage of records´ representations

Connectivity	1 Gbps
Access	through the two hybrid nodes, acting both as storage and access points (proxies)
Hardware	physical machines, for each node: 2 CPUs (6 cores, 2 GHz), 32 GB RAM, 9 TB HDD

NoSQL Database Cluster

Size	6 machines
Software	Apache Cassandra 2.0.11
Function	Storage of technical metadata
Connectivity	1 Gbps
Access	peer-to-peer cluster, any node is access point
Hardware	physical machines, for each node: 2 CPUs (6 cores, 2 GHz), 64 GB RAM, 2.6 TB HDD

Computational Cloud

Indexing Cluster

Size	2 machines
Software	Apache Solr 4.5.1
Function	Search index over technical metadata
Connectivity	1 Gbps
Access	master-slave cluster, reads via any node, writes via master
Hardware	physical machines, for each node: 2 CPUs (4 Cores, 2.4GHz), 48 GB RAM, 265 GB HDD

Messaging Cluster

Size	2 machines
Software	Apache Kafka 2.9.2-0.8.1.1
Function	Messaging system (Europeana Cloud Asynchronous Messaging Service)
Connectivity	1 Gbps
Access	leader-follower cluster, reads via any node, writes via partition leader[2]
Hardware	physical machines, for each node: 2 CPUs (4 Cores, 2.4GHz), 48 GB RAM, 265 GB HDD

Logging and Monitoring Cluster

Size	1 machine
Software	Ganglia 3.6.0/LucidWorks Banana 3/Apache Solr 4.5.1
Function	Logging Applications (Europeana Cloud Logging Service)
Connectivity	1 Gbps
Access	one node only
Hardware	virtual machine, 2 virtual CPUs (2.4GHz), 2 GB RAM, 40 GB HDD

Frontend Services Cluster

Size	3 machines
Software	Apache Tomcat 7/Apache httpd 2.2 Europeana Cloud Services, Apache Zookeeper 3.4.
Function	Frontend Services of Europeana Cloud: Metadata and Content Service Unique Identifiers Service Data Lookup Service

[2] http://kafka.apache.org/documentation.html#introduction

	Authentication-Authorization Service
Connectivity	1 Gbps
Access	any node
Hardware	virtual machines, for each node: 6 virtual CPUs (2.4GHz), 6 GB RAM, 50 GB HDD

Security

It is increasingly important to ensure security of information systems in the face of threats of cyber attacks on political, economical, religious or other grounds. Especially for distributed environment, as Europeana Cloud, with multiple components and entry points, it is advised to hold regular assessments of overall robustness of the system against potential attacks. To address this need, the Europeana Cloud installation in PSNC, described above, was thoroughly tested by the PSNC Security Team. The installation and configuration of the operating system as well as of the application container (Apache Tomcat) and the firewall (Apache Httpd) were tested on all the machines that will be publicly accessible. The audit focused on components that are known to be vulnerable according to PSNC security considerations. The system successfully passed the inspection except for a few minor problems in configuration of Apache Httpd and Apache Tomcat. Those problems were resolved.

To further ensure high security of the Europeana Cloud services, the system should also be inspected on the level of code beyond the test done on the network mentioned above, which is to be tested for fault and security threats tolerance. This kind of security audit should be performed in further project stage.

Performance

We performed a suite of performance measurements to assess the performance of the system. At this stage we measured the performance of the two user-facing services (MCS and UIS) as the main entry points for data. In the future, dedicated measurements should also be performed on the backend services.

Measurement Setup

The measurements were performed using tests run by Apache JMeter. Five separate JMeter test cases were prepared: (1) UIS, (2) MCS DataSet, (3) MCS Representation 1MB, (4) MCS Representation 5MB, (5) MCS Representation 10MB. A single test session consists of six test iterations with variable number of threads (from 1 to 100).
Tests cases where run one after another with thirty seconds delay. In a test case there was one second delay between each request thread group. Every request was performed 10 times by each thread from a thread group.

The tests were run from a single machine which has 6 core CPU, 2 GB of RAM, 1 Gbps network card and runs Ubuntu 12.04.5 LTS.

Software used in tests:

- testing tools: JMeter 2.11, Java 1.7.0_40-b43
- monitoring tools: Sysstat 10.0.2-1, Nload 0.7.3-1.0, Htop 1.0.1-1

The tests were ran against the Metadata and Content Service (MCS) and the Unique Identifiers Service (UIS) deployed in a single application container (Apache Tomcat) on an application cluster machine. For security reasons on application machine Apache Httpd server is used. It passes traffic to application container. Currently there are three endpoints:

- directly to Tomcat server - 8080
- Apache Httpd http port - 80
- Apache Httpd https port - 443

There is 1 Gbps connection between the machine running the services and the testing machine.

For gathering statistics we were using Ganglia monitoring software, installed on all the machines. During the test, the system had only one proxy node of Openstack Swift cluster. The seven remaining nodes where cluster nodes. The Apache Cassandra cluster had 5 nodes where available.

Test Description

The test run consisted of six test iterations, each one with variable number of threads. Each iteration emulates a group of simultaneously working users when the number of users increases to track the dynamics of system´s performance. The table below lists the number of threads in each iteration and the its length.

Number of threads	Start time	Finish time
1	11:22:36	11:26:34
20	11:28:36	11:34:22
40	11:36:23	11:45:00
60	11:47:02	11:58:21
80	12:00:22	12:14:43
100	12:16:44	12:33:51

The table below represents average times for basic operations achieved during the test. The response times for twenty threads (means twenty users making operations in parallel) were relatively low. Further increasing number of threads caused longer time processing of single request.

Request type / Number of threads	Average request response time [ms]					
	1	20	40	60	80	100
assign Representation to DataSet	24	46	208	291	255	294
create CloudId	16	17	15	15	17	17
create DataSet	48	82	382	526	454	405
create Representation	39	140	445	501	1035	1142
delete DataSet	150	160	250	361	370	457
delete Representation	22	84	443	661	673	855
get all versions of record Representation	8	61	125	238	511	722
get CloudId	18	62	129	91	256	272
get DataSet	141	149	192	235	295	344
get file from Representation	32	61	112	223	410	553
get Representation	12	37	200	296	282	363
list all latest persistent versions of record Representation	20	70	145	176	548	772
persist Representation	26	132	275	574	1007	1397

Figure 3 shows the usage of CPU of the application machine. As the number of threads increases, the CPU consumption grows. But the growth is not linear, indicating that the application is not entirely CPU-bound. The CPU consumption itself is within the expected range.

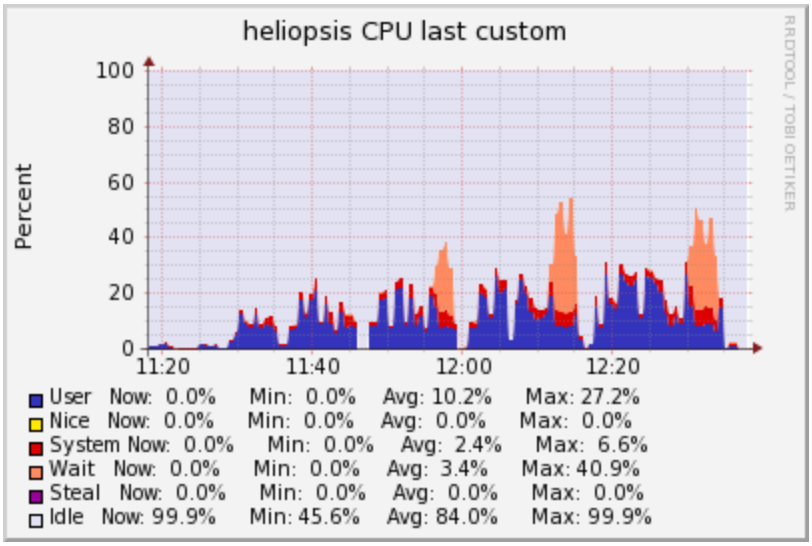

Figure 3: Application machine - CPU usage

FIgure 4 shows load on the application machine. There is a significant load increase during uploading of large files. OS was scheduling processes in a way, that new tasks where waiting in a queue for available resources. So, new requests must wait in a longer queue and the application response time was longer. The issue will be investigated further.

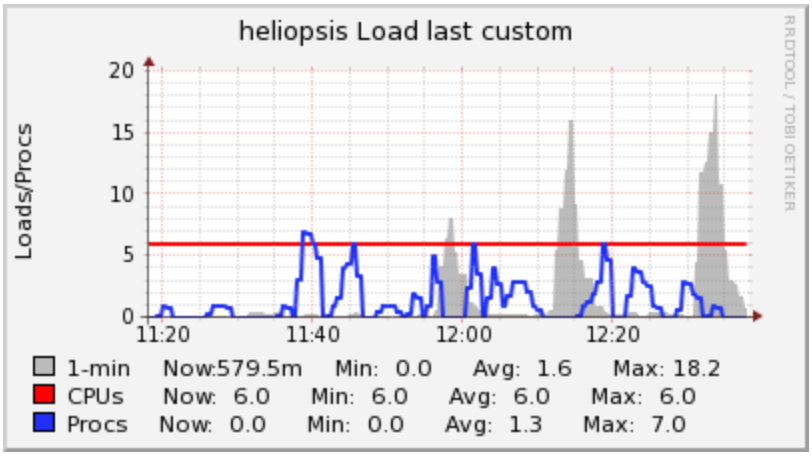

Figure 4: Application machine - load

Figure 5 shows network load on the application machine. The maximum load (about 45 MBPS) was reached during the test of 60 threads. While we would expect linear growth, the load did not increase further. Our working assumption is that this behaviour is explained by an internal limitations of OpenStack and we shall investigate it further. Statistics for incoming and

outgoing traffic are similar. That can point that all the data is transferred to the backend and Europeana Cloud applications are handling most of the test requests properly.

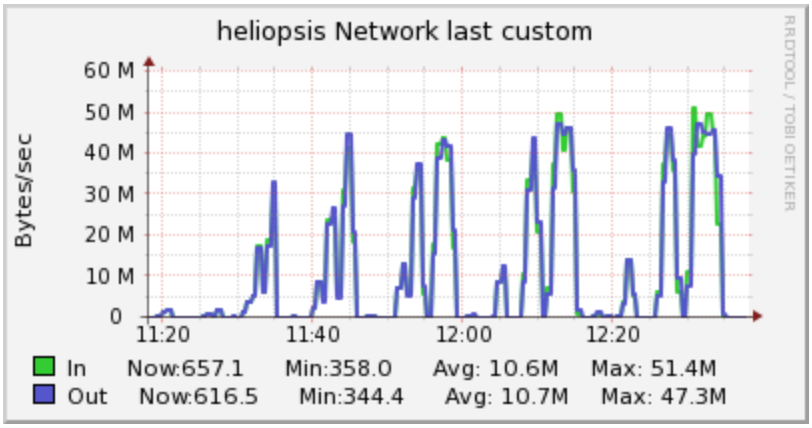

Figure 5: Application machine - Network usage

Memory usage of the application machine stayed constant on the expected level during the entire test. Cached memory increased during uploading large files.

Backend Services

Monitor backend systems is as important as monitoring user-facing application machines. Below we present performance statistics gathered on the two clusters of the Storage Cloud. We show both cumulative graphs showing aggregate statistics for all the cluster´s machines. These should be indicative of the condition of the entire cluster. The network summary report includes incoming traffic from the outside of the cluster and also the traffic inside of the cluster.

NoSQL Database Cluster

During the test Apache Cassandra cluster load peaked at 5.7% and 1-minute sliding average load was low at 0.8%. Used memory remained stable and low. This indicates that a lot of computing resources were idle.

Figure 6 shows cumulative network usage of the Apache Cassandra cluster. Increasing number of threads results in linear increase of the overall cluster network usage. The increasing amount of data coming from the application results in a increasing outgoing traffic (nodes are communicating with each other). There is no saturation effect.

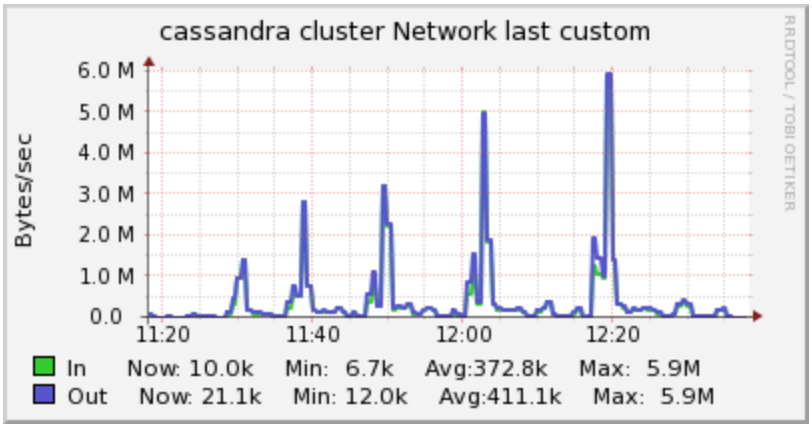

Figure 6: NoSQL Database Cluster - Cumulative network usage

Figure 7 shows network usage on each node of the cluster. All nodes of the cluster manage traffic at comparable level. Load profile is consistent with the cumulative one so there is no favouring of any node.

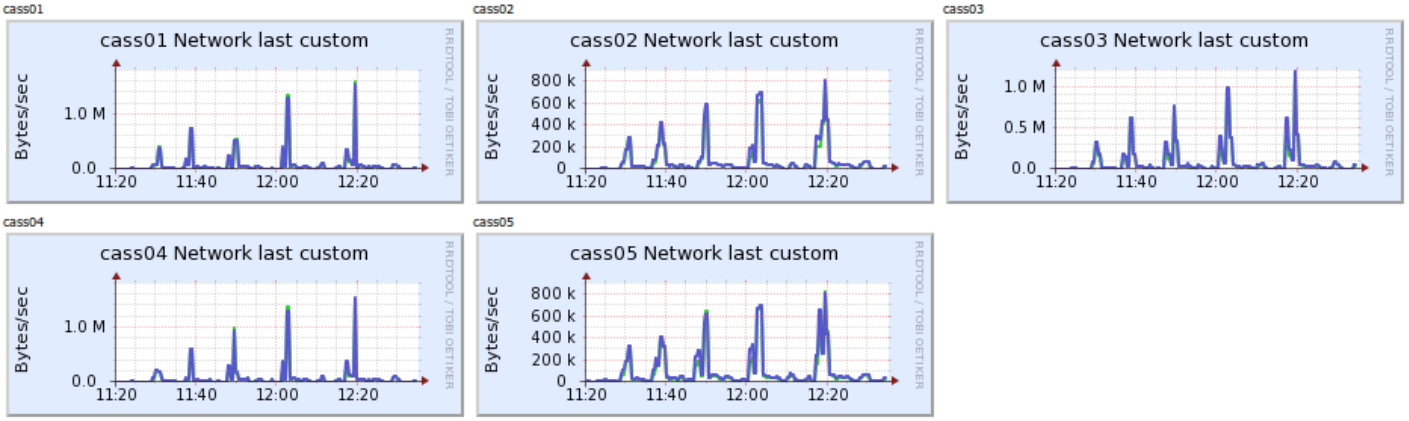

Figure 7: NoSQL Database Cluster - Network usage per node

Object Storage Cluster

During the test the CPU load was low: it peaked at 6,4% and the sliding 1-minute average was negligible. Also memory usage was stable and low. A lots of computing resources were idle.

Figure 7 shows cumulative network usage of the Object Store Cluster. Aggregate traffic got saturated during the 40-threads iteration. For incoming traffic maximum of 311.3 MB/s is reached. For outgoing traffic it is 264.4 MB/s. There is a correlation between traffic on the cluster and on the application machine. For some reason, after a certain level of parallel request the average request processing time increased due to a higher load from application

machines. There is also a disproportion between incoming and outgoing traffic. This is because one of the nodes is a proxy node that transfers data to storage nodes and this traffic is also included in this cumulative chart.

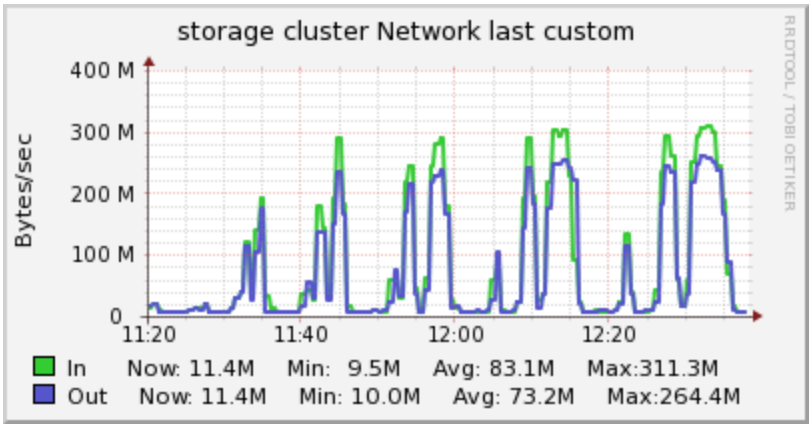

Figure 7: Object Store Cluster - Cumulative network usage

Figure 8 shows OpenStack Swift network usage on each node of the cluster. Node 10.21.24.129 is a hybrid node that is a gateway to the Swift cluster and transfers the traffic to the storage nodes. Saturation (reaching a limit) is specifically visible on outgoing traffic from the hybrid node.

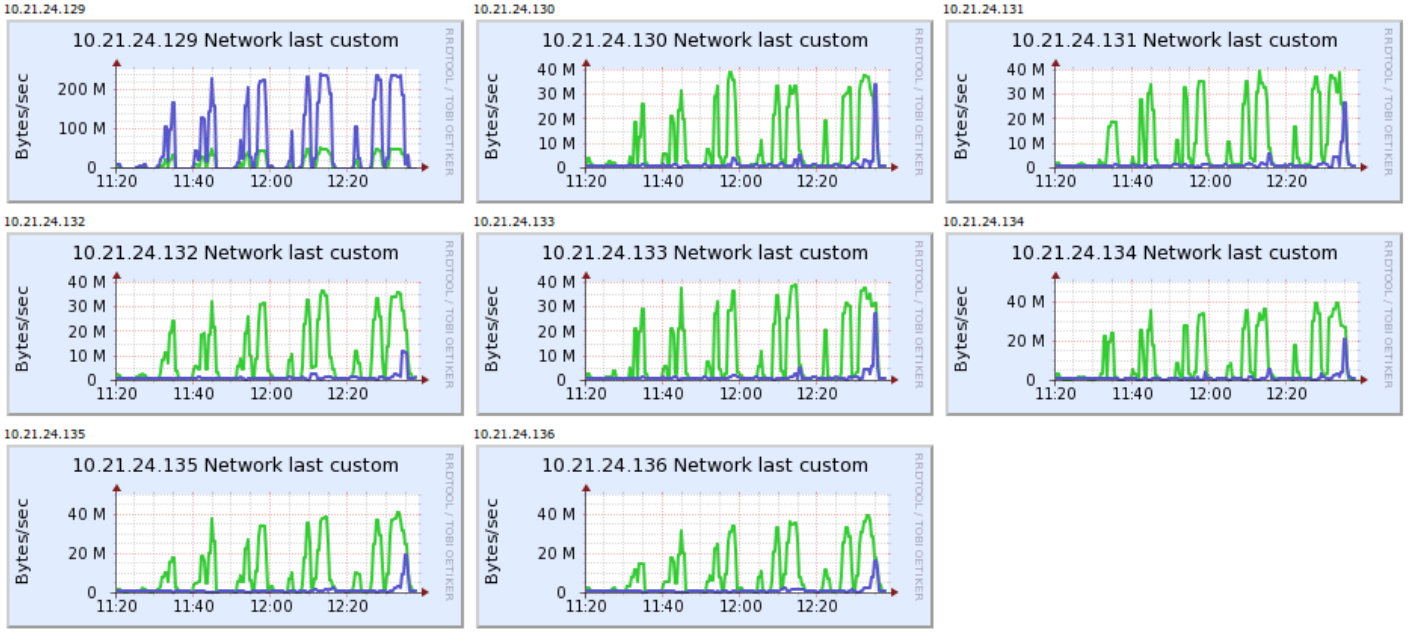

Figure 8: Object Store Cluster - Network usage per node

Pilot Data Migrations

Clepsydra

The Clepsydra/Europeana Cloud batch import application was developed to transfer data from the Clepsydra storage system, used by the Polish Digital Libraries Federation to Europeana Cloud.

Every record stored in Clepsydra consists of two main parts: header and data. A header contains all the record´s metadata. A data is a file (usually in xml format) which contains metadata about stored (digitized) object and binary streams of the object. When different representation of the same digital object are stored in Clepsydra, they are stored as separate records with the same id (fbcId) but with different values of header values and data. In essence, the Clepsydra data model is a flat model. Records do not form datasets.

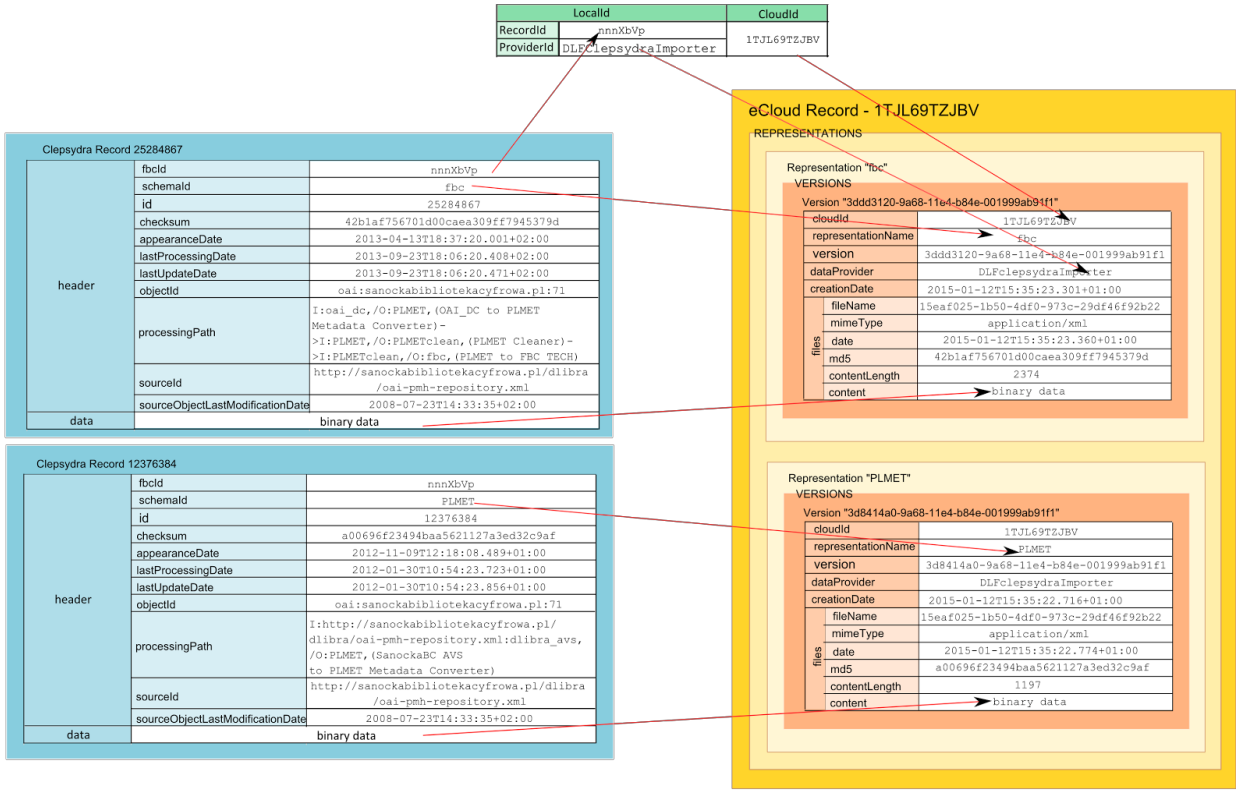

Figure 9: Mapping between the Clepsydra data model and Europeana Cloud data model

Figure 9 illustrates how two different Clepsydra records with the same fbcid are migrated to create two representations of a Europeana Cloud record. Because Clepsydra does not have

the notion of a dataset, all the resulting Europeana Cloud records were bound into the same dataset.

Therefore, after the import process, the following new objects are created in Europeana Cloud:

- All Clepsydra records having the same Clepsydra ID (*fbcId*) will correspond to different representations of the same Europeana Cloud record. An entry containing this *fbcId* and the Europeana Cloud ID of the record will be added to the mapping table
- Clepsydra schema ID (*schemaId*) field will be used for representation name in Europeana Cloud records
- All the new Europeana Cloud records will belong to the same provider identified with this import procedure
- The content of digital objects stored in the *data* field of Clepsydra records is copied in the *files* part of Europeana Cloud Records

The import application was executed at a single thread on a development machine. The test took about 10 hours. During that time 200K records was transferred. This limit was due to the performance limitation of the Clepsydra system and of the machine on which the test was ran. We have not reached the limitation of Europeana Cloud performance during this test. However, not all the aspects of the system were tested, such as search, for example.

Our takeaway from this experiment is that there are two kinds of tasks when importing data from another system to Europeana Cloud: modeling and implementation. Modeling takes time because peculiarities of the source and the target data models have to be understood in order to create the mapping. Implementation is made simple by a clear and accessible import interface of Europeana Cloud. It is desirable for the source system to have a clear and accessible export interface as well, as in Clepsydra´s case. Lastly, it is important to note that when time estimate of a migration process is done, performance limits of the source, the target and the migration machines should be taken into consideration.

Open University

The CORE/Europeana Cloud migration was developed in order to transfer scholarly metadata from Open Access Repositories (CORE) to Europeana Cloud.

CORE stores different pieces of data related to the same aggregated record. These parts are stored in Europeana Cloud as representations of the same record. A record can contain:

- XML (OAI-PMH) representation of the original metadata record. This representation can have multiple versions reflecting changes done over time to the original record.
- JSON representation that contains content metadata and information about record aggregation timestamps and status in the system

- full text (PDF) of the record (optional)
- raw text format of the record (optional)
- thumbnail preview of the first page of full text (when full text available)

The entire migration application is still under development but after its completion it will synchronize in total 20.6 million records (corresponding size of 6.7 TB) from CORE to Europeana Cloud as a part of the regular harvesting update procedure of the system.

At the moment, the migration application is single-threaded and its performance is mostly bounded by the network throughput when transferring data. But since logical operations, such as creating records and representations are also time-consuming, we expect the overall efficiency to improve in a multithreaded application.

There are two takeaways from this experience. First, batch operations will be useful for such massive data migrations. Second, clients who want to keep their local repositories synchronized with Europeana Cloud will need to store locally (meta)data about the synchronisation process, which can be costly.