Institute of Parallel and Distributed Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Master Thesis Nr. MCS-0002

Adding Value to Object Storage: Integrating Analytics with Cloud Storage Back ends

Hoda Noori

Course of Study:	Computer Science
Examiner:	PD Dr. rer. nat. habil. Holger Schwarz
Supervisor:	Dipl.-Inf. Tim Waizenegger
Commenced:	1. Nov 2015
Completed:	2. May 2016
CR-Classification:	E.1, E.2, H.2.4

Abstract

With the vast interest of customers in using the cloud infrastructure, cloud providers are going beyond limits to offer advanced functionalities. They try their utmost best to present the services in a way that makes the customers highly attracted and convince them about value and benefits of using such services. For this purpose, cloud providers need to have an access to customers' data, hence customer-sensitive data stored in repositories should be transferred to the cloud. Object storages are one of the possible solutions for the implementation of repositories in cloud environments. However, due to the data being confidential and fragile, security and encryption mechanisms are required. The application of Enterprise Content Management (ECM) system highly relies on metadata, thus there is a need to keep metadata unencrypted while encrypting data itself. Therefore, cloud providers that are hosting ECM systems are forced to keep metadata unencrypted in order to satisfy the main functionalities of ECM systems on the cloud. Although other cloud providers can offer data encryption and unencrypted metadata as an option to their customers. This leads to the conclusion that enhancing object storages with analysis capabilities in ECM systems is more beneficial if it is done on top of unencrypted metadata. In this thesis I investigate how value can be added to such cloud storage services by only using access the metadata. I specifically focus on providing analytics functionality on metadata.

This Master's thesis aims at providing the means to efficiently analyze the metadata inside a cloud-based ECM system (OSECM) which uses Swift Object Store as its back end repository. I extended the OSECM system with required components by providing new modules that enable the retrieval of metadata from the object storage and the insertion of this metadata into a metadata warehouse. The importance of metadata replication in a distinct data warehouse offers the possibility of benefiting from SQL query capabilities for analysis purposes. Furthermore, an existing tool was integrated as the analysis component to offer the means for interaction with the underlying metadata warehouse and the user interface. Finally, after applying analysis queries, the results are presented on the user interface using the predefined set of visualization interfaces. The supported data structure for the visualization of the result are also defined in this work.

Contents

1 Introduction 9
- 1.1 Motivation . . . 10
- 1.2 Background . . . 11
- 1.3 Related Work . . . 24

2 System Architecture and Components 31
- 2.1 Swift Object Store . . . 34
- 2.2 Content Identifier . . . 35
- 2.3 Metadata Extractor . . . 36
- 2.4 Metadata Replicator . . . 36

3 Analysis Concepts 39
- 3.1 Visualization . . . 41
- 3.2 Candidate Use cases . . . 46

4 Technology and Implementation 49
- 4.1 How does Swift work? . . . 49
- 4.2 Content Identifier . . . 51
- 4.3 Metadata Extractor . . . 51
- 4.4 Metadata Replicator . . . 51
- 4.5 DataModel . . . 52
- 4.6 Bluebox UI . . . 54
- 4.7 Analytics Diagrams . . . 55
- 4.8 SQLite Database . . . 69

5 Summary and Future Work 71

Bibliography 73

List of Figures

1.1 Data Science Metromap [Cha13] . . . 12
1.2 Data Science Pipeline [Oje14] . . . 13
1.3 Data Wrangling using TRIFACTA [**TRIFACTA**]. . . . 15
1.4 A modern Data Architecture with Apache Hadoop integrated with existing data systems [Hor14] . . . 21
1.5 Enterprise Hadoop Components [Hor14] . . . 22
1.6 Potential Use Cases for Big Data Analysis . . . 25
1.7 Alfresco Repo Architecture . . . 26
1.8 Alfresco Analysis Architecture . . . 27
1.9 Alfresco Analytics Flow of Data . . . 28
1.10 Nuxeo Rep Arch . . . 30

2.1 The OSECM Architecture [**OSECM**] . . . 32
2.2 The process of extracting metadata from objects . . . 33
2.3 Swift Data Model; includes Accounts, Containers and Objects plus their attached metadata . . . 35
2.4 The Metadata Replication Workflow . . . 38

3.1 The required components and architecture to create the analysis layer; Considering that object store and Bluebox UI is existed . . . 40
3.2 The analysis scenarios inside Node-RED Server; providing an Endpoint for outside access . . . 41
3.3 Different part of the Bluebox UI . . . 42
3.4 The Component Overview of Bluebox User Interface [**OSECM**] . . . 43
3.5 The BB-Insights User Interface inside Bluebox . . . 44
3.6 The supported data structure for the visualization . . . 45

4.1 The Metadata Extractor Diagram . . . 52
4.2 The Metadata Replicator Class Diagram . . . 53
4.3 The Metadata warehouse schema in SQLite database . . . 54
4.4 The Analytics Component Interactions . . . 56
4.5 The Content Type Numbers and Size Query . . . 57
4.6 The Content Type Distribution Bar Graph . . . 59

4.7 The Line Graph of the Object Distributions based on their Size; The query is mentioned in figure **??** 61
4.8 The Area Graph of the Content length Distribution 62
4.9 The Stacked Bar Graph of the Content Type Size and Number Distribution; It uses the same dataset as figure **??** which is the result of query in the figure **??**. 63
4.10 The Group Bar Graph of the Content Type Size and Number Distribution; It uses the same dataset as figure **??** which is the result of query in the figure **??**. 64
4.11 The Box Plot of the Size and Number Distribution grouped by the Object Content Type; It is the result of query in the figure **??**. 65
4.12 The Donut Chart of the dataset in figure 4.3 67
4.13 The Bar Chart to show the result set for the query in figure **??** 68

List of Tables

4.1 The content type distribution data set 58
4.2 The data set of the object distributions based on their size. It is the result for the query of figure **??**. 60
4.3 The data set for the distribution of medals in various countries 66

List of Listings

4.1 The Content Type Distribution Query 57
4.2 The query to expose the object distribution based on their size 58
4.3 The Content Type Numbers and Size Query 59
4.4 The query to fetch the distribution of size and number of documents for each content type. 61
4.5 The query to fetch the email addresses of people who has sent the largest amount of email. The top 20 senders is queried to have a better visualization in the bar graph. 66

4.6 The query to fetch the list of people who are communicating with each other via email. 67

List of Algorithms

1 Introduction

Many companies use Enterprise Content Management (ECM) systems to organize and archive their various kinds of data. Usually the stored data inside ECM systems are huge as they are including several documents for companies like invoices, contracts, project documentations, reports, scanned letters, emails and other documents. These documents are usually stored in the ECM systems for archiving, categorizing and managing the company data with considering security aspects as well. As Waizenegger et al. is mentioned, in order to benefit from diverse advantages of cloud-based systems like improving the quality of the service and reducing the cost, many enterprises have a desire to migrate their legacy applications including the Enterprise Content Management system to the cloud environment [**OSECM**].

One major advantage of cloud services is that the providers can continuously improve their available services and also offer additional services to customers; like analytics on data that is already stored on the cloud. Cloud applications are usually built as combination of multiple small components which are independent and loosely coupled, thus in order to migrate the legacy systems into the cloud, it is better to split the whole system to the small parts and use the best practices to move these specific parts to the cloud. Data repository is one of the main parts for each application and becomes more important for those systems which deal with a large amount of data. Therefore, moving the repository to the cloud environment can bring more benefit for organizations including the cost reduction of providing enough storage for the system at each time. Object storage is one of the best cloud-based storage solutions because of its scalability and elasticity which allow companies to store as much data as they need during the time. Furthermore, object storage provides a convenient way to access data via HTTP APIs.

Object storages have another use case besides cloud storage which is being used in the content management systems as well. Therefore here I focus to provide an analysis layer for an existed ECM system (OSECM) which is using an object storage as its repository. This analysis layer brings the ability of finding out more hidden insights about the structured big data which is stored in the object storage.

First I investigate data science various domains, tools and technology and present some of the existed solutions and frameworks for big data analysis. Then, I introduce a way to provide the analysis layer via employing some open source applications and engines

that can be reused and easily integrated with the existed ECM solutions. In addition, some new components should be designed and developed in order to integrate this layer with the existed ECM system. These new components perform fetching the desired data, storing it in specific storage, applying analysis queries and visualizing the results. Nevertheless this existed ECM system itself is based on a framework that provides main required components prototype and details on how they should be realized to form a cloud native ECM system.

1.1 Motivation

Since companies are dealing with a large collection of data in their different applications that they have, including Enterprise Content Management system, they prefer to outsource their data storages to the cloud providers to reduce their cost and benefit from scalability and elasticity of cloud services. Using the cloud infrastructure enables the cloud providers who keep the data, to offer extra services like data analysis. Usually ECM systems utilize object storages as their cloud-based repository when they want to migrate to the cloud environment. Particularly in ECM systems, since we need to store and utilize both binary data and metadata, object storages are one of the good choices to be used as a repository in cloud-based ECM systems. With the possibility to keep a considerable amount of metadata using object storages, searching, mining, data protection and analytics can be done more efficiently within the cloud-based ECM system.

In fact, since the ECM systems use the metadata for managing and organizing their contents, metadata (e.g. object name, object size and others) is a key element in these systems to find and have an access to object. In addition, having these structured and relational metadata fields for objects in ECM system, enables cloud providers to analyze this metadata information rather than using unstructured and more complicated object content data. Here, in the OSECM system, the metadata is also organized and structured through defining all document classes and content type categories which exist in the system. The other reason that in this work, the system uses metadata for analysis is because of the fact that data needs to be encrypted in the cloud infrastructure. Normally when organizations want to store their data in the cloud environment, they prefer to have enough security for their information thus they request cloud providers to provide the encryption facilities for their data. However, many cloud providers do not offer the encryption services or they only support a restricted encryption services to be able to sell other services to their customers with using their actual data.

In the OSECM system, the offered compromise is that the encryption is only applied for the data itself and not for the metadata. Therefore, this is another reason that why

here I provide analysis feature for the metadata and not the object data itself. Although user can still restrict the metadata fields to fulfill enough security level because without having some metadata unencrypted the system becomes almost unusable. For instance, if the name of the object is encrypted then the object can not be found anymore. In this case, user can ask for the object through giving the name of object to the cloud provider and get the object, decrypt it and then actually read and use it. Here, I also attempt to make the required new components for analysis targets, in a way that they can be simply integrated with the existed ECM system. Since other components inside this ECM system are designed in a loosely coupled architectural style, I prepare separated small modules with specific functionalities defined for analyzing the data too. In this way there is enough flexibility to replace most of these components with existed solutions that bring the same functionality.

Here, the OSECM system uses Swift Object Store from OpenStack[1] project as its repository which brings several advantages including data protection, cost and scalability.

Object storages are providing enough metadata to ease security, optimization and data resiliency processes, while offering an indexed file structure that's ideal for unstructured data and allows files to be distributed over wide geographic areas with a minimized effect on performance.

1.2 Background

Generally Big Data is a generic term to address the large volume amounts of structured and unstructured data that is collected from multiple sources and continuously flows through and around organizations. This data can provide answers to questions they may not have even thought to ask but extracting the useful data from big data is a main challenge which cause the emerge of data science.

In fact, in the area of data science, it is discussed that in each and every matured organization how can understand it's business and generate some results to optimize the business. In order to reach this goal, there are lots of various tools, technologies, articles and domains which are used to collect data, clean and integrate it, apply various analysis and modeling techniques and finally represents the results to exploit the generated profitable data for fulfilling the desired business targets [Cor15a].

Statistics, Machine Learning, Big Data Analytics, Text Mining and Natural Language Processing, Data Storage Mechanisms, Visualization techniques are some of the main

[1] http://docs.openstack.org/developer/swift/

debated domains in Data Science. **Chandrasekaran** in his work perfectly pictured almost all the skills, technologies, tools, programming languages and libraries, databases and storage systems which are needed to be learned to become a data scientist [Cha13].

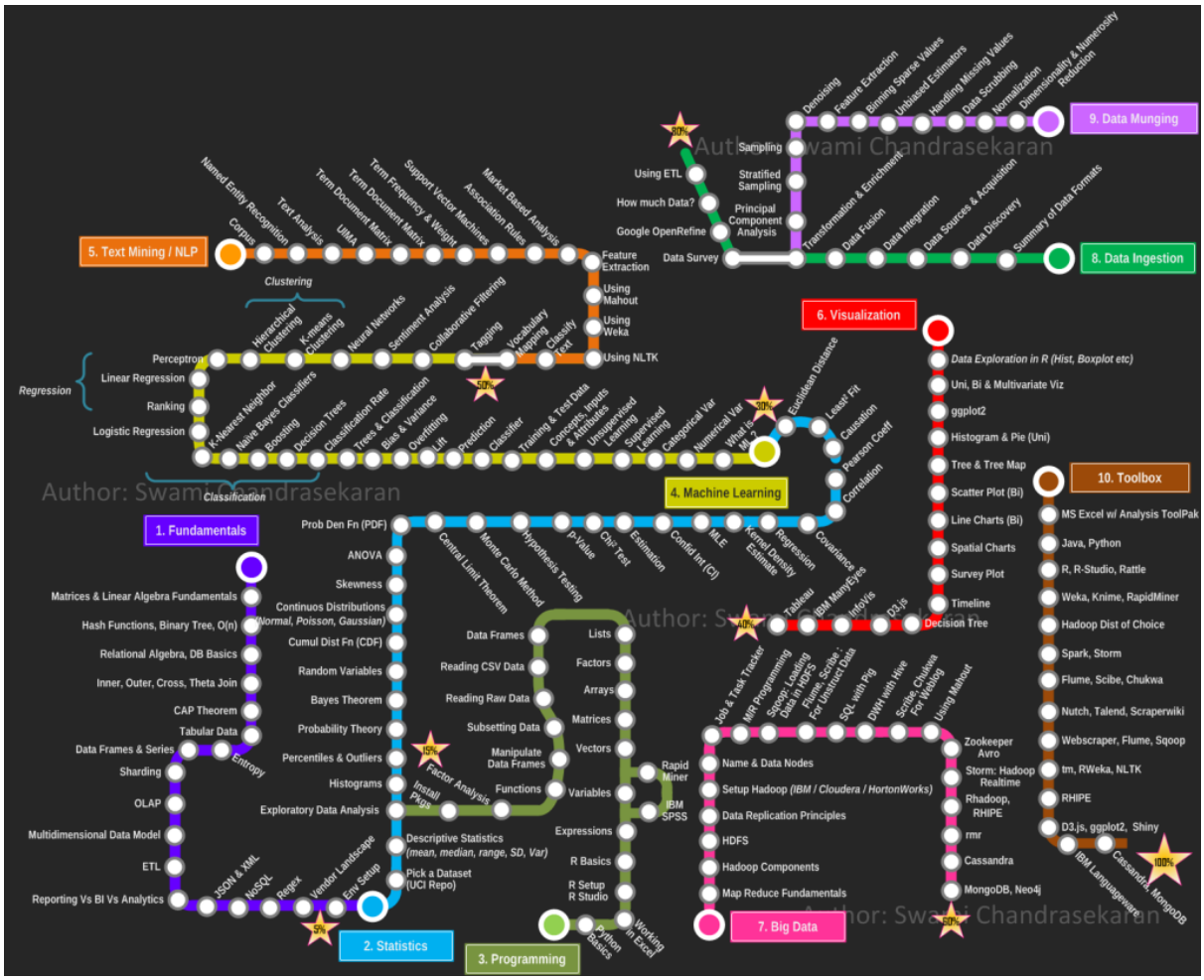

Figure 1.1: Data Science Metromap [Cha13]

Generally in order to analyze data, some functions and operations should be performed which are categorized in various steps that are referred as Data Science Pipeline in literatures. These information is presented in the figure 1.1 as well. Although this pipeline does not have the same steps for each and every use cases and some of them might be skipped in different cases. Mainly Data Science pipeline consists of the following groups of steps which is also presented in the figure 1.2.

1. Data ingestion; Data acquisition and recording.
2. Data munging and wrangling; information extraction and cleaning [Oje14].

3. Data integration, aggregation, and representation.
4. Query processing, data modeling, and analysis [DE+12].
5. Interpretation; Reporting and visualizing.

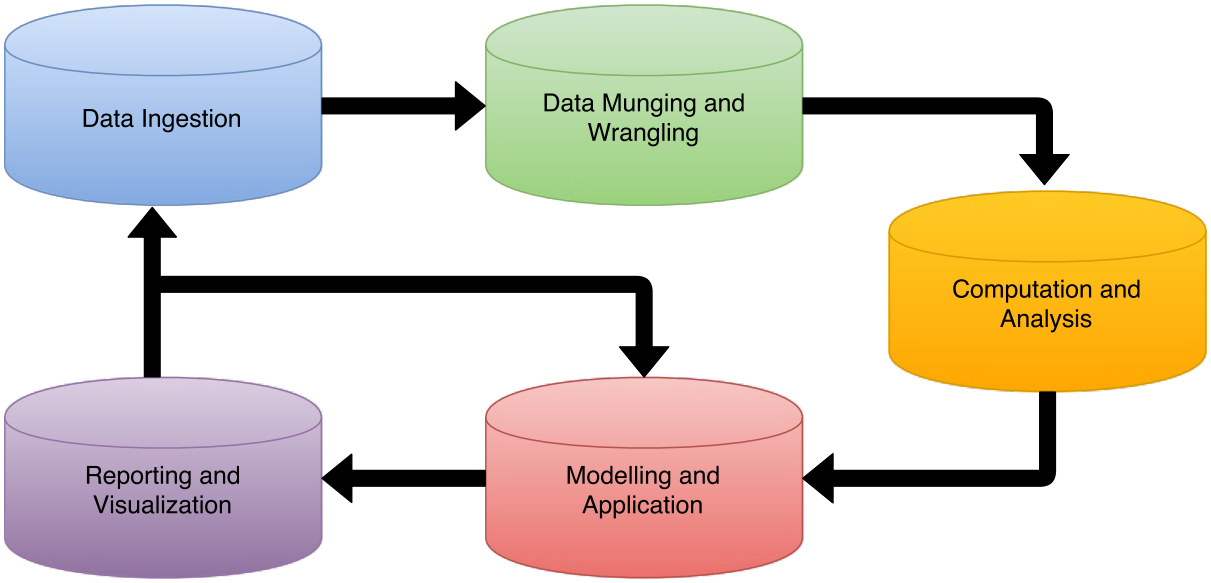

Figure 1.2: Data Science Pipeline [Oje14]

1.2.1 Data Ingestion

Big data comes from various data generating sources. Integrating these various sets of data is an essential requirement for making sense of available big data and putting it to the productive use. So the process of accessing, importing, and processing data, storing it in a database or in a storage system for later usage is called Data Ingestion.

As Swami Chandrasekaran also has mentioned in his Data Science Clock, there are a couple of challenges which should be considered in this step [Cha14]. Since a large number of conducted data might not be useful, the first challenge is to define some filters such that the useful information would not be discarded. Hence we need a smart process to reduce the size of original data without missing the significant parts. Moreover in some cases there should be some online analysis techniques to process the streaming data lively because for these cases like embedded sensor data, it is not possible to save all information and process it afterwards [DE+12].

The second challenge is to generate the correct metadata automatically to define which data is stored and how it is archived and sized. Meta data acquisition system can

decrease the amount of effort to keep metadata. The other considerable point is to extract the origin of the data and keep this information through data analysis pipeline. For example in the case that we have an error during one step the processing phase, it might cause some following analysis faults that are linked to this step. Hence generating convenient metadata and having a data system to carry the root of data and its metadata along the data analysis pipeline should be taken to the account in this step of data science pipeline.

IBM[2] is one of the sources that introduced two functional categories for Data Ingestion phases: batch and streaming. Ingesting batch data includes fetching data in distinct blocks, like transactions dumps of a day. Streaming ingestion, also named real time ingestion, means every data record is ingested distinctively as it is evolved by the source, like sensor monitoring systems or social network messages. Gobblin, Amazon Kinesis, Apache Samza, Cloudera Morphlines, White Elephant, Apache Chukwa, Heka, Databus, Apache Sqoop, Apache Flume, Scribe and Fluented are some of the data ingestion tools without any specific order[3].

1.2.2 Data Munging and Wrangling

Most often all the collected information is not relevant to the addressed problem, in addition, it might not be in the desired format ready to be analyzed. Thus, we need an information extraction process to select the right information from the primary resources and represent it in the structured format which is appropriate for analysis [Cor15a]. Furthermore, sometimes data extractions for images and videos are depended to the related application. For instance, data extraction requirements of a MRI is obviously different than a map or a supervision photo [DE+12].

Thus, cleaning and formatting data known as “Data Munging and Wrangling” are the most time-consuming steps in the data science pipeline. In real world analysis, data wrangling can consume up to 80% of project time. The Extract/ Transform/ Load (ETL) process can be used by professional data scientists to clean and prepare data sets for analysis. In fact, *data wrangling* is referred to transforming and converting the data from a primary format into a target format which consists of extra munging, data visualization, data aggregation, providing a statistical model and other possible applications. To employ *data munging*[4], the next general steps should be fulfilled: first the primary format of data should be fetched form the data source, then applying the

[2] IBM Big Data and Analytics Hub : http://www.ibmbigdatahub.com/blog/ingesting-data-data-value-chain

[3] 12 Data Ingestion Tools: http://www.predictiveanalyticstoday.com/data-ingestion-tools/

[4] https://en.wikipedia.org/wiki/Data_wrangling

required algorithms like sorting the data and transforming it into the specified format and lastly keep the result data set inside a data sink for future applications.

The importance of *data munging* in the data science domain is crucial. The data quality determines the level of complexity in order to fulfill data munging process.

DataWrangler

Stanford-Berkeley research project provided an interactive tool for purifying and reconstructing the data [**DataWrangle**]. Particularly, this tool is designed to improve the process of fetching data to be analyzed and visualized and decreasing the required time to read the data. It allows the user to find out more information about data along with the interactive it is possible to intract utilizing this tool, Using such a tool allows you to spend more time to learn about data as well as interactive manipulation of unstructured and dirty data to prepare it for the analysis usages.

The *DataWrangler* research project is finished and currently the software is not supported. Instead, **Trifacta Wrangler** is other commercial tool which efficiently prepares various data to be used in analytics or visualization tools such as *Tableau* [**TRIFACTA**].
The figure 1.3 presents the data wrangling using Trifacta.

The main Trifacta Wrangler benefits are the followings:

- Improving the analysis process.
- Easing of data preparation process.
- Supporting more diverse and complex set of data to be analyzed.

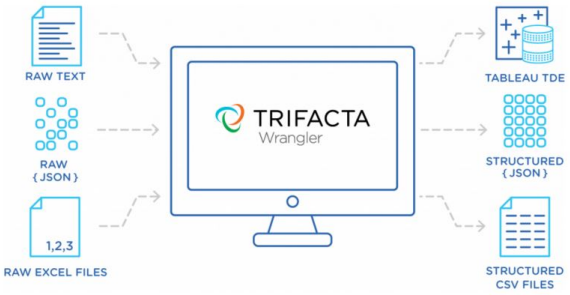

Figure 1.3: Data Wrangling using TRIFACTA [**TRIFACTA**].

1.2.3 Data integration, aggregation, and representation

Normally it is not a good approach to put a bunch of raw data directly into repositories because in this case finding the desired information and reusing it will be hard. Keeping metadata along with the data can slightly help but still there are some challenges related to the practical details and structure of data records.

Analysis of data is profoundly more challenging than only locating, recognizing, realization and observing data [DE+12]. In fact, all of these steps should be performed in a fully automated manner in order to analyze data efficiently. Therefore, data structure needs to be translated in order to be systematically understandable for computer. Data integration can be one of the useful solutions although it requires extra effort to make differences clear in an automated error-free way.

Furthermore, for the easy analysis cases, it is vital to design database appropriately. Most often there are different alternatives to how information can be recorded. Specific design have some benefits for predefined targets and probably some disadvantages for others. Hence, now a days, database design is a vital skill in the enterprise context that is done by special experts. There are some creative tools which help other nonspecialist experts, like domain scientists, to design databases effectively [DE+12].

1.2.4 Query Processing, Data Modeling, and Analysis

Custom statistical analysis on small data instances are essentially different from querying and general Big Data analysis approaches. Most often Big Data is dirty, unreliable, heterogeneous, correlative and unsafe. Even though messy Big Data could be more worthwhile than small samples since typical statistics achieved from repeated patterns and interrelation analysis usually overcome specific variations and often reveal more trustworthy hidden models and observations. Moreover, missing data can be inspected with information redundancy that existed in the Big Data and large incoherent information structures. In addition, investigating inconsistent cases, approving reliable inter connections, displaying substantial clusters, and detecting invisible relationships and models are possible as well.

In order to apply data mining, it is required to have organized, trustable and efficient data set along with descriptive query and appropriate mining interfaces and algorithms as well as required computing environment. Moreover, data mining can improve the quality of the data and supply more complicate querying functionalities. The real-time analysis is the next group of interactive data analysis for Big Data. On the long run, queries can be systematically generated on the insertion of a new content in a website,

produce recommendations and offer specific analysis to make a decision on top of a data set to either store or discard it.

Currently, Big Data analysis has a problem in the lack of communication between data storage systems which keeps the data and generate SQL with analysis applications which require non-SQL operations, for example, data mining or statistical analyses [DE+12].

General Analytical Tools and Technologies

There are several well-known tools and technologies which are used for analysis purposes in industry. Some of them are open source and for the others there are commercial licenses which specifies the level of provided features.

R Language: R is a system for statistical computation and graphics. It consists of a language plus a run-time environment with graphics, a debugger, access to certain system functions, and the ability to run programs stored in script files [R C16].

RapidMiner: RapidMiner (formerly known as YALE), offers advanced analytics through template-based frameworks. It has a profound advantage that users rarely have to write any code. It is offered as a service, rather than a piece of local software and is one top rated data mining tools. Moreovre, RapidMiner also provides functionality like data preprocessing and visualization, predictive analytics and statistical modeling, evaluation, and deployment[5]. What makes it even more powerful is that it provides learning schemes, models and algorithms from WEKA and R scripts.

WEKA: WEKA[6] originally was developed for analyzing data from the agricultural domain. With the Java-based version, it is very sophisticated and used in many different applications including visualization and algorithms for data analysis and predictive modeling. It is free under the GNU General Public License, which is a big advantage compared to RapidMiner, because users can customize it however they prefer. WEKA supports several standard data mining tasks, including data preprocessing, clustering, classification, regression, visualization and feature selection.

[5] https://rapidminer.com
[6] https://weka.wikispaces.com/

KNIME: KNIME supports all data preprocessing steps which are reading, transformation, analyzing and deploying. It provides a graphical user interface to assemble nodes for data processing. It is an open source data analytics, reporting and integration platform. In addition, KNIME integrates various components for machine learning and data mining through its modular data pipelining concept and supports business intelligence and financial data analysis as well.

Statistical Analysis System (SAS): SAS is a software suite developed by SAS Institute which is the leader in analytics. It provides innovative analytics, business intelligence, data management software and predictive analytics[7]. In fact, it is a software system for data inspection and report writing. Statistical Analysis System is a batch of programs that work together to reclaim them and to reserve data values, reform data, evaluate complex and single exponential analyses and generate reports[8].

IBM Watson Explorer: IBM Watson Explorer is a perceivable investigation solution that integrates search and content analytics which helps users to discover and comprehend the required in order to work more thoroughly and produce more precise result to make decisions [Cor15b].

1.2.5 Interpretation; Reporting and Visualizing

After getting the raw data from various data sources and manipulation it, it is important to understand the data so that later we can interpret that data through visualization. We should transform the data into the format which can be leveraged by visualization tools. knowing the best way to take the data and visualize it in different possible diagrams like tree maps, link graphs and parallel coordinates, is the other issue which should be considered. Visualization theory, dashboards, Visualization tools (e.g. Gephi, Mandrian, Afterglow, twopi) and libraries have to be considered as well.

Visualization Tools and Technologies

Visualizing data is important regardless of the size of the data because it translates information into insight and action. The approach to visualizing Big Data is specially important because the cost of storing, preparing and querying data is much higher. Thus,

[7] http://www.sas.com/
[8] https://intellipaat.com/tutorial/statistical-analysis-system-sas-tutorial/

there are many tools and technologies available, each with their different strengths. to visualize the data in order to reveal more hidden insights and information. Tableau, Excel Chart Animation, Data Exploration in R, Ggplot2, Apache Zeppelin[9], ZoomData[10], D3.js, InfoVis, IBM ManyEyes, QlikView, Plotly[11].

1.2.6 Data Storage and Management Technologies

It is one of the main challenges in the Big Data area is that how can we securely store a large amount of data like capture log files and even contexts.

Relational Databases

Data with the following characteristics might be better suited for a traditional RDBMS: 1.OLTP required (On-Line Transaction Processing) 2.ACID requirements (atomicity, consistency, isolation, durability) 3.Complex data relationship 4.Complex query requirements [**DataModel**].

NOSQL Databases

Data with the following characteristics is well-suited for a NoSQL system: 1.Data volume growing rapidly 2.Columnar growth of data 3. Document and tuple data 4.Hierarchical and graph data [**DataModel**].

File System and Data lakes

The term data lake comes from the big data community and starts appearing in the security field more often. A data lake (or a data hub) is a central location where all security data is collected and stored. Sounds like log management or security information and event management (management or security information and event management: SIEM) [Mar16]. A data lake is a large storage repository and processing engine, they provide "massive storage for any kind of data, enormous processing power and the ability to handle virtually limitless concurrent tasks or jobs" [Wik].

[9] https://zeppelin.incubator.apache.org/
[10] http://www.zoomdata.com/
[11] https://plot.ly/python/overview/

1.2.7 Technologies for Big Data

There are couple of tools and technologies which are used in the area of data science. MS Excel with Analysis Toolpack, Java, Python, JavaScript, R, R Studio, Rattle, Weka, Knime, RapidMiner, Hadoop Dist of Choice and Apache Spark are some of well-known tools in this area.

Apache Hadoop

Apache Hadoop is a one of the well known frameworks in the Big Data area. It is a scalable error prone and distributed system for repository and processing of large collection of data sets. Hadoop can be used to keep large of unstructured and semi-structured data safely on several clustered servers while scaling performance considering being cost effective by slightly inserting cheep nodes to the cluster [Vik14]. Elastic search and log stash and know how to make useful visual representation of your data.

The following areas are various capabilities of Enterprise Hadoop which are a main requirement for each platform and technology:

Data Management Supply and process large collection of data in a scalable repository layer.

Data Access Read and communicate with the stored data in multiple ways.

Data Administration and Integration Rapidly and simply load data, and manage based on policies.

Security Fulfill the need of Authentication, Authorization, Accounting and Data Protection.

Operations Provision, manage, monitor and operate Hadoop clusters at scale [Hor14].

The detail of Apache projects is presented in the figure 1.5 which satisfy all the mentioned functionalities.

Data Management: Hadoop Distributed File System (HDFS) is the main technology for the efficient scale out storage layer, and is designed to run across low-cost commodity hardware. Apache Hadoop YARN is the pre-requisite for Enterprise Hadoop as it provides the resource management and pluggable architecture for enabling a wide variety of data access methods to operate on data stored in Hadoop with predictable performance and service levels.

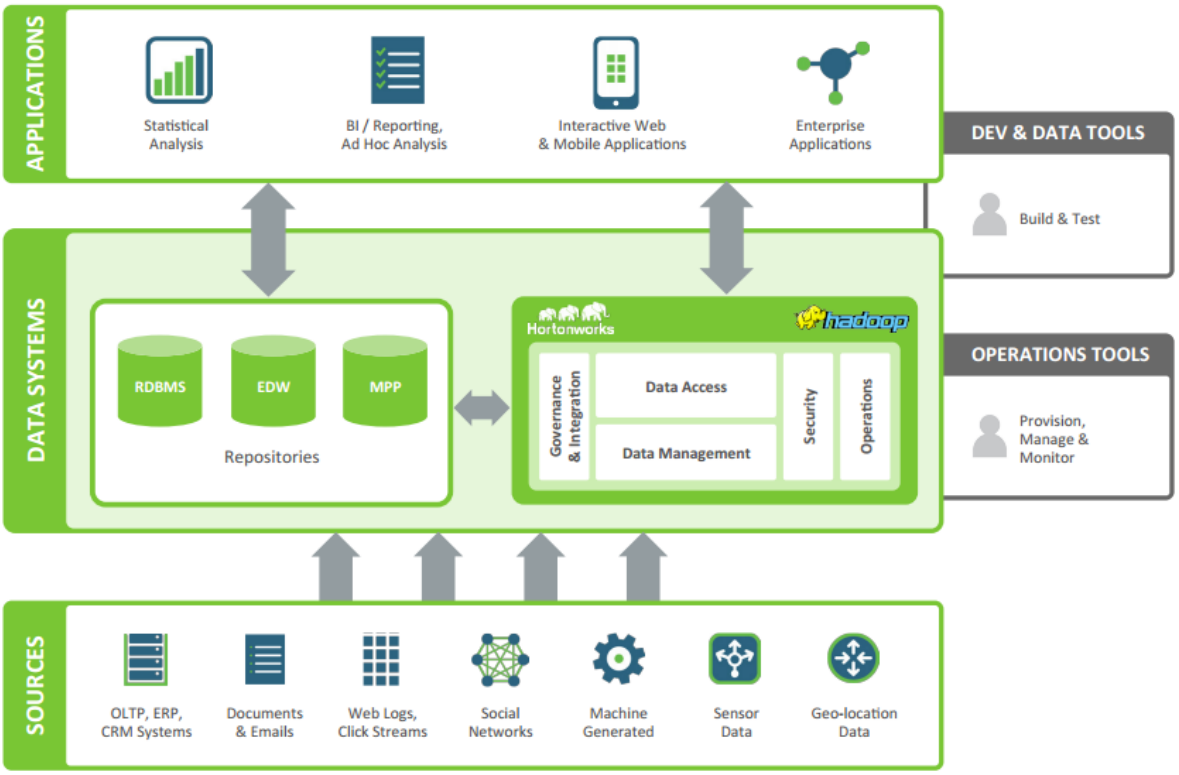

Figure 1.4: A modern Data Architecture with Apache Hadoop integrated with existing data systems [Hor14]

Data Access: Apache Hive is the most accepted data access technology, although there are multiple specific engines. For example, Apache Pig offers scripting functionalities, Apache Storm provides real-time operations, Apache HBase creates columnar NoSQL storage and Apache Accumulo provides cell-level access control. All of these parts can employ on one set of resources. Moreover, YARN offers flexibility for advanced methods of programming frameworks.

Data Governance and Integration: Apache Falcon offers policy-based workflows for governance, while Apache Flume and Sqoop offers plain data ingestion same as the NFS and WebHDFS interfaces to HDFS.

Security: Security is offered for all layers in Enterprise Hadoop.

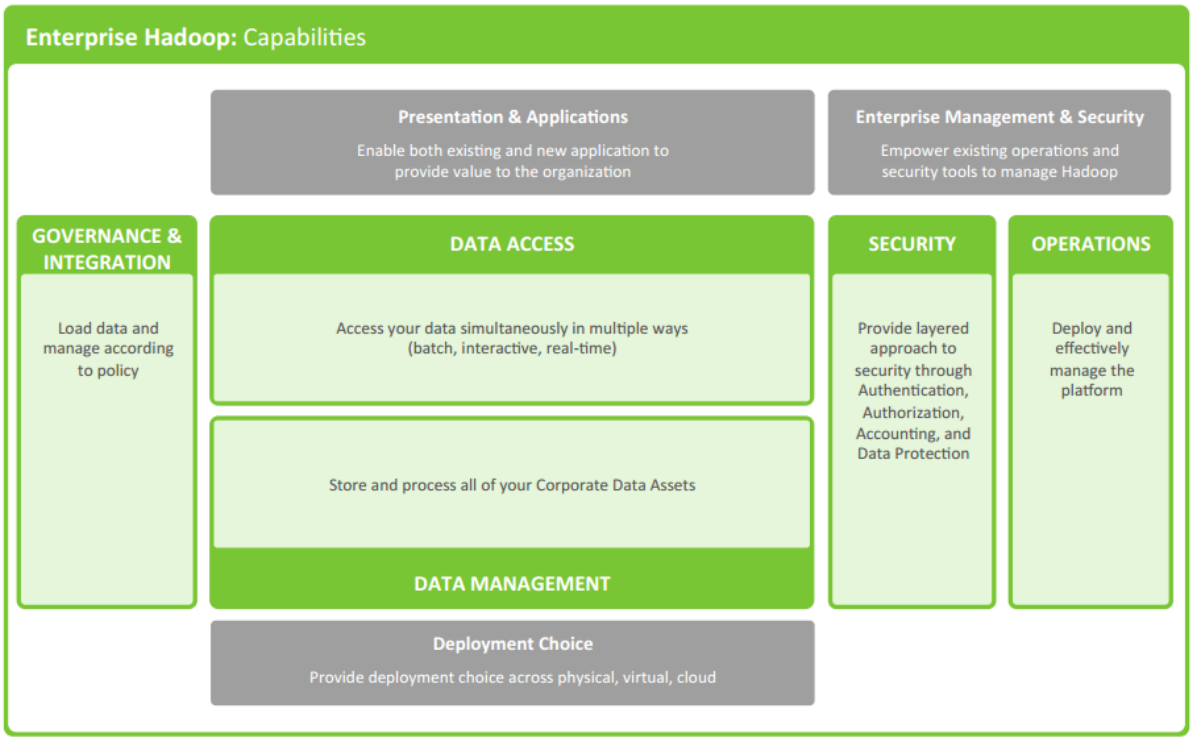

Figure 1.5: Enterprise Hadoop Components [Hor14]

Operations: Apache Ambari provides the necessary interface and APIs to supply, organize and control Hadoop clusters and merge with other management software [Hor14].

1.2.8 Object Storages

Object storages are used as a file server for cloud-based applications. They provide extended features for companies in the case that they have an application which should be run on the cloud. For instance, some web based applications like content management system, usually have a relational database to store metadata related to the contents. In the past, the content file itself were stored in the local file server which needs server configuration to replicate the file server. Since the accessibility of file servers is hard on the cloud, they are not anymore used on the cloud environment. In this case, the object store is a replacement for file server, for example, if there is an ECM application as a cloud-based application which is run on the cloud and includes a set of image data set, thus, the images in this system should be stored inside a object storage. Due to the fact that object storages have a **REST** based API and all the programming languages can

be connected to them, they are good choices to be used a data storage on the cloud. Therefore, if local file servers are replaced by large object storages, the target application is ready to be run in the cloud environment. Moreover, object storages could be used as a cloud-based back end storages for Enterprise Content Management systems as well. There is a possibility to use an object storage as a general purpose data store specifically for analysis targets similar to Hadoop Distributed File System (HDFS)[12] which is mostly used to store the data for analysis purposes.

Migration Reasons to Object Storages

Traditionally many companies have big servers with file servers installed on it which are very hard to maintain. Because they have all these folders which different people access to it and both structured and unstructured data are always stored together. Since object storages are technically easy to manage, many companies which have a large installation, want to migrate to the object storage. In case of having a large installation, companies have to install all the servers, although with the object storages usage, it is easy to maintain and operate these servers beside having a good user interface.

Object Store Advantages

Totally using object storages has some advantages and disadvantages which are listed in the followings.

- Scalable capacity
- Scalable performance
- Persistent
- Low cost
- Simple management
- Single access point
- No volumes to manage and change the size[13].

[12] https://hadoop.apache.org/
[13] Online Resource: http://blog.rackspace.com/introduction-to-object-storage/

Object Store Disadvantages

- POSIX utilities do not work directly with object-storage since it is not a file system
- Integration may need changing of application and work flow logic
- Usually, lower performance on a per-object basis than block storage[14].

Data and Object Storage Scenarios

Generally there are various potential use cases for Big Data analysis that are mentioned in the figure 1.6. For some of these scenarios, due to the some reasons, it is a good idea to use the object store to keep the Big Data on the cloud.

There are couple of various use cases which object store is appropriate with their requirement as a storage rather than file systems. In general different areas can be targeted to apply analysis methods in order to gain the target output which could be generated in various kinds of formats. In the following there is a list of different data types that can be stored inside object store with the list of possible metadata that they can have. Email Archiving is one good example to implement.

Enterprise Content Management (ECM) is another good use case of using object stores. Business records, invoices, contracts and other company specific data are stored in ECM systems. like that. Other examples are cloud applications, websites, online applications and similar applications that contain various types to store objects. Object storages are more used as back end in cloud-based applications and due to extensive support of metadata, they are a goof choice for a repository of the ECM systems on the cloud.

1.3 Related Work

There are different solutions and technologies that provide a possibility to merge analytics features with the existed data management system running on the cloud and then visualize the result in a convenient way. many companies use the Enterprise Content Management System (ECM) to maintain and manage all their documents, datum, business application contents or multimedia assets and most of them prefer to use the open source solutions. Since most often, there is a huge amount of data in an ECM system, it is preferable to apply some analytics scenarios on top of the data to find more details

[14]Online Resource: http://blog.rackspace.com/introduction-to-object-storage/

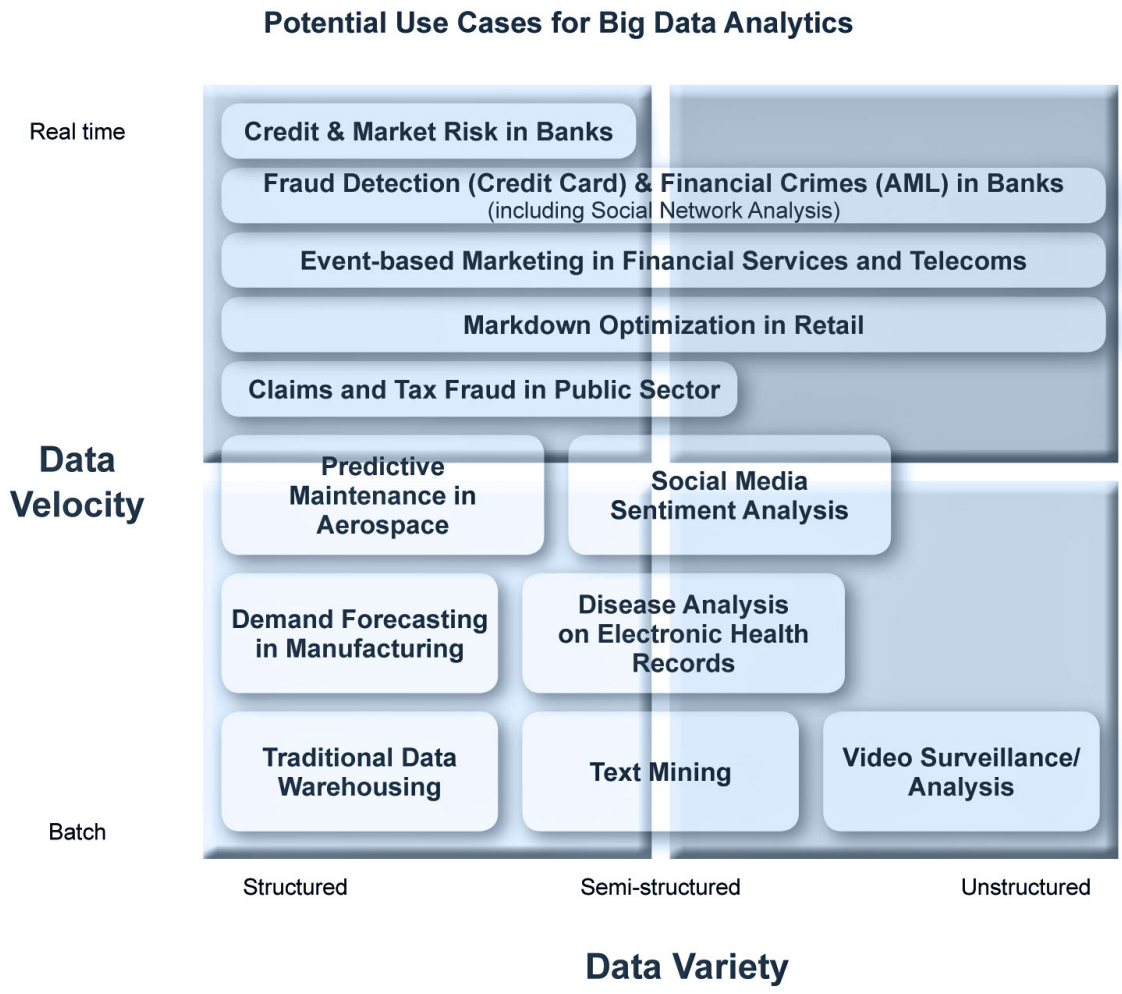

Figure 1.6: Potential Use Cases for Big Data Analysis
[15]

via visualizing the result or providing reports. For this purpose, there are some open source ECM platforms that support analysis features as well. Alfresco[16] and Nuxeo[17] are two sample platforms which both are built in a Fine-grained and modular approach of architecture just like the OSECM system therefore it is easy to extend, customize and integrate them with the existing applications and processes. However, Alfresco Enterprise version is closed source, its Community version is open source. Nuxeo is a real open source model because code is available for the Enterprise version too [DB13]. Alfresco is leading the combination of ECM and Business Process management (BPM)

[16] https://www.alfresco.com/
[17] http://www.nuxeo.com/

to make efficient connected processes which present content in context. Moreover, "it provides easy mobile access to content, delivers a simple but rich collaboration user experience and helps customers maximize the value of their content[18]". Since Alfresco has an integrated analytics component, it is easily possible to access and composite data from Alfresco in order to obtain beneficial insights, and make information-driven decisions[19]. The integrated analytics extends exploration by finding content and interactions besides detecting old assets to archive. Alfresco provides customizable filters to refine searches by document metadata in sort of property, format, content author, site, tag and more automatic extraction and indexing of metadata and in this case it is similar to the OSECM system. As it is clear in the figure 1.7, similar to the OSECM system, Alfresco stores metadata in a database and content in a file system. Content is kept in the file system to provide for very large content, random access, streaming and options for different storage devices.

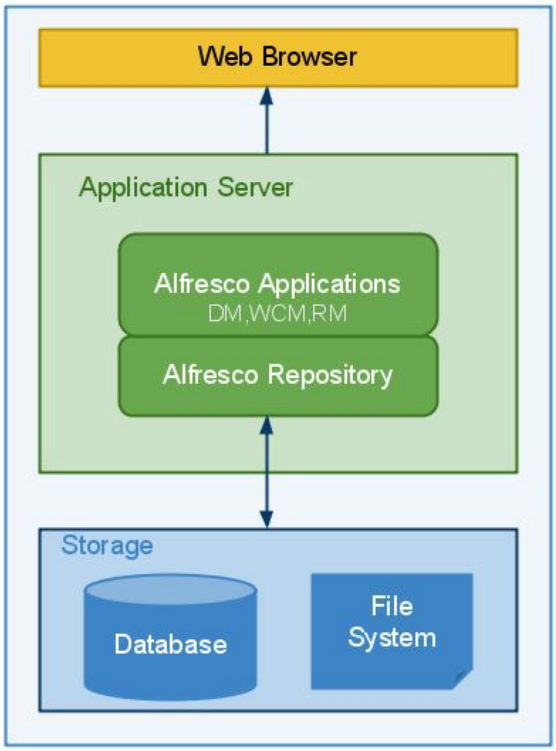

Figure 1.7: The Alfresco Repository Architecture[20]

[18] https://www.alfresco.com/products/enterprise-content-management
[19] http://docs.alfresco.com/analytics/concepts/analytics-overview.html

The Alfresco architecture provides a good flexibility to substitute the components with others that can meet the same functionality.

Alfresco analytics provides more visibility over Alfresco sites, user and contents via providing reports. It mines the information of user activities and new contents in order to produce these reports. The provided reports are either pre-defined or custom reports. A collection of interactive reports based on users and content creators are offered as pre-defined reports, for example, the most active users. The business requirements can be fulfilled through producing custom reports specifically for that business purpose. These reports have different level of securities. They can be published publicly into the Alfresco websites or maintained privately.

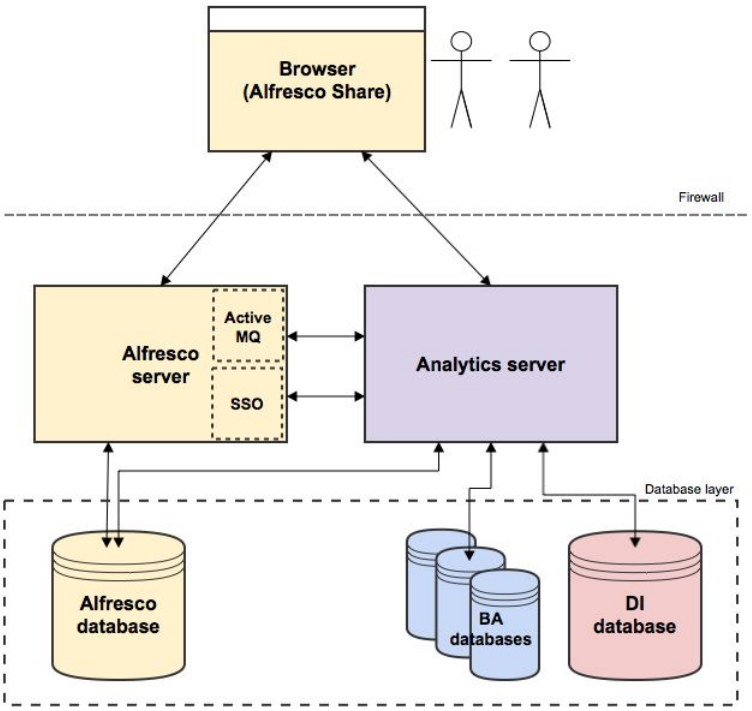

Figure 1.8: The Alfresco Analytics Architecture[21]

[20] https://wiki.alfresco.com/wiki/Alfresco_Repository_Architecture
[21] http://docs.alfresco.com/analytics/concepts/analytics-architecture.html

As it is shown in figure 1.8, the Alfresco Analytics consists of a set of components. The business analysts can access to the reports through "Alfresco Share" component. These reports are generated based on the events that are caught and stored in the database. When an event happens, a flow of events take place in order to provide the data that can be queried by a Business Analyst to make reports. The figure 1.9 shows the flow of data that occurs in Alfresco Analytics when an event happens. For instance, when a user uploads a new document through Alfresco Share, the document is stored in Alfresco repository. This event is captured by ActiveMQ queue as an message and is kept in the Data Integration (DI) database. Through ETL (Extract, Transform and Load) process, the data inside Alfresco database is transformed and sent to the DI database which communicated with the Business Analytics (BA) server. The report definitions are saved in the BA database. Finally, a business analyst can produce and edit reports through Alfresco Share component.

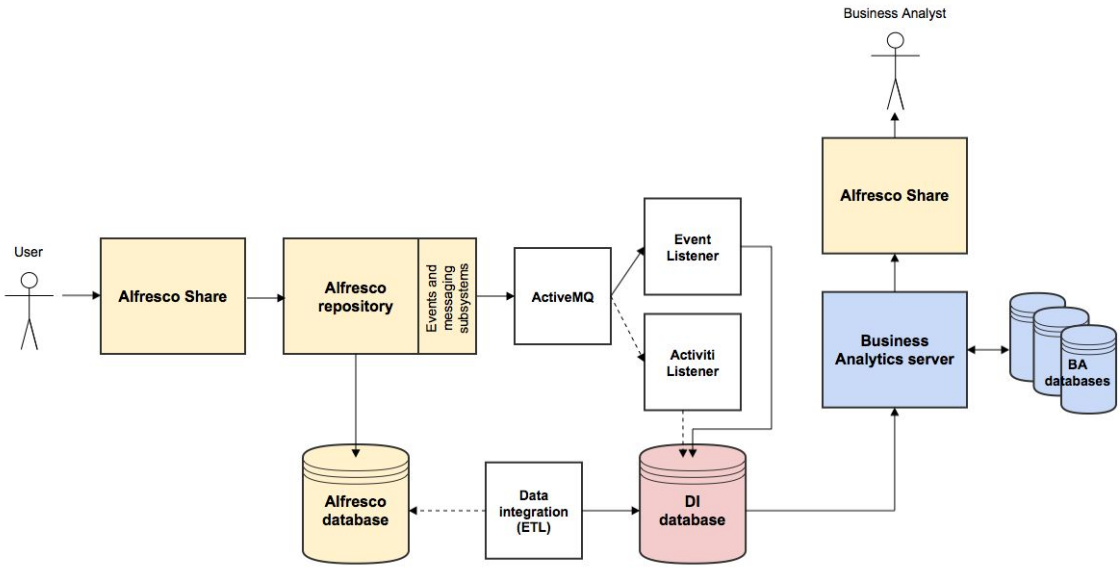

Figure 1.9: The Alfresco Analytics Flow of Data[22]

[22] http://docs.alfresco.com/analytics/concepts/analytics-detailed-architecture.html

In compare with both Alfresco and Nuxeo platforms, in the this work, the system provides a facility for user to easily define the analysis queries and change them through using NODE-RED server which is an interactive tool. This facility enables user to write SQL queries with various range of complexity and even use JavaScript functions to further processing data within JavaScript code to restructure the data to make it compatible with the supported visualization interfaces in the Bluebox UI system.

The Nuxeo Platform is a developed platform for building modern content applications. It is extremely modular and has more than 150 modules to support all its capabilities[23]. Similar to Alfresco and OSECM system, the Nuxeo Platform has the repository for storing content and associated metadata. It stores metadata in a SQL database structure while the content binaries are usually on a file system[24]. As it is shown in the figure 1.10, these parts in the Nuxeo repository are communicating with Document Store and Blob Store respectively.

The document store is implemented by the visible content store (VCS) and any SQL database. While PostgreSQL is preferred for performance reasons, any SQL database can be plugged if user prefers. All metadata is stored in the VCS and a Document is referred to a collection of metadata which the binary or blob is only one field of a Nuxeo document. The blob store is normally based in a file system and a simple structure where binary content is renamed with a unique ID. This is the only reference to the VCS. Since it has a flat and simple architecture, it does not need complex file system operations and allows switching to other blob managers such as Amazon S3. Nuxeo provides a flexible and extensible REST API for complex interactions and can integrate with many cloud service to create, query and manage content using the well-known applications. For instance, Nuxeo Vision[26] is one of its services which provides an integration with Google Cloud Vision image detection service[27] to derive insight from images with the powerful Cloud Vision API.

Generally, as I discussed in the background chapter, there are various open source data analysis tools and frameworks that some of them could be used for the analysis component in this work as well.

[23] https://doc.nuxeo.com/display/NXDOC58/Architecture
[24] http://www.nuxeo.com/blog/nuxeo-architecture/
[25] 24
[26] http://www.nuxeo.com/nuxeo-vision/
[27] https://cloud.google.com/vision/

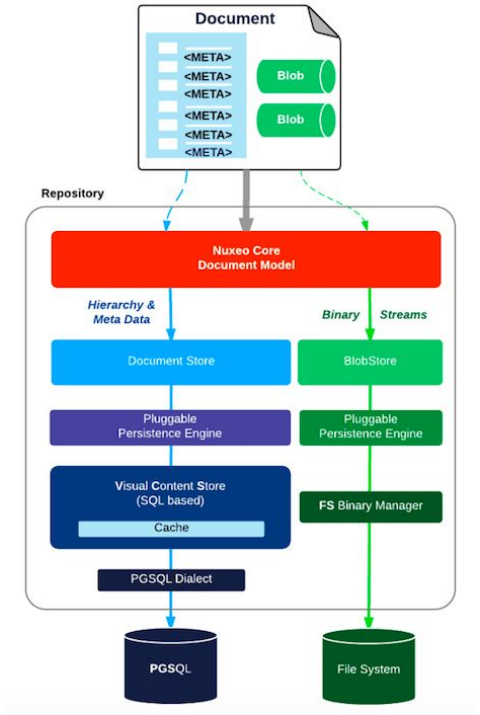

Figure 1.10: The Nuxeo repository architecture[25]

2 System Architecture and Components

This work focuses on creating an analysis layer for an ECM system which is running on the cloud and is using object storages as its back end repository. I started with a cloud-based Enterprise Content Management (ECM) system which is using Swift Object Store from the OpenStack project[1] as its main storage. Swift is one of the cloud-based object storages.
The existed Enterprise Content Management system itself is based on a framework which targets creating a prototype to support both creating new cloud native ECM components and help to migrate the existed ECM systems to the cloud environment [**OSECM**]. However, in this work, I attempt to enhance the OSECM system with analysis capabilities. For this purpose, new components are designed in a way that can be integrated with the existed system easily and can be applied to existed ECM systems or other applications which are relied on cloud object storages as well. The architecture of the OSECM system is pictured in the following:

As it is shown in the above picture, in OSECM system all features has been divided to a group of small components with specific functionalities. There are three data management systems in the whole OSECM system which are used for different purposes: Message Persistence, Swift Object Store and Metadata Warehouse. The Swift Object Store and Relational Metadata Warehouse are involved to create an analysis layer, that I will describe the flow of their usage in details later in this chapter. In order to benefit the advantages of object storage features, specially for cloud-based applications, Swift Object Store is used in the OSECM system. Typically it is used for storing large binary objects and wide amounts of objects and in OSECM system, it is considered as a file system for this cloud-based application.
In the OSECM system, I focused to create an analysis layer for all objects that have been stored inside the Swift Object Store in order to distinguish the hidden insights and knowledge about them through using their associated metadata. Generally every object inside ECM systems has the actual object data plus metadata which describes the information about the object. In Swift, there are metadata for all the different types of entities that it has and all of them are stored as key-value pairs. In OSECM system also metadata provides the possibility to classify, advanced search and analysis objects.

[1] http://docs.openstack.org/developer/swift/

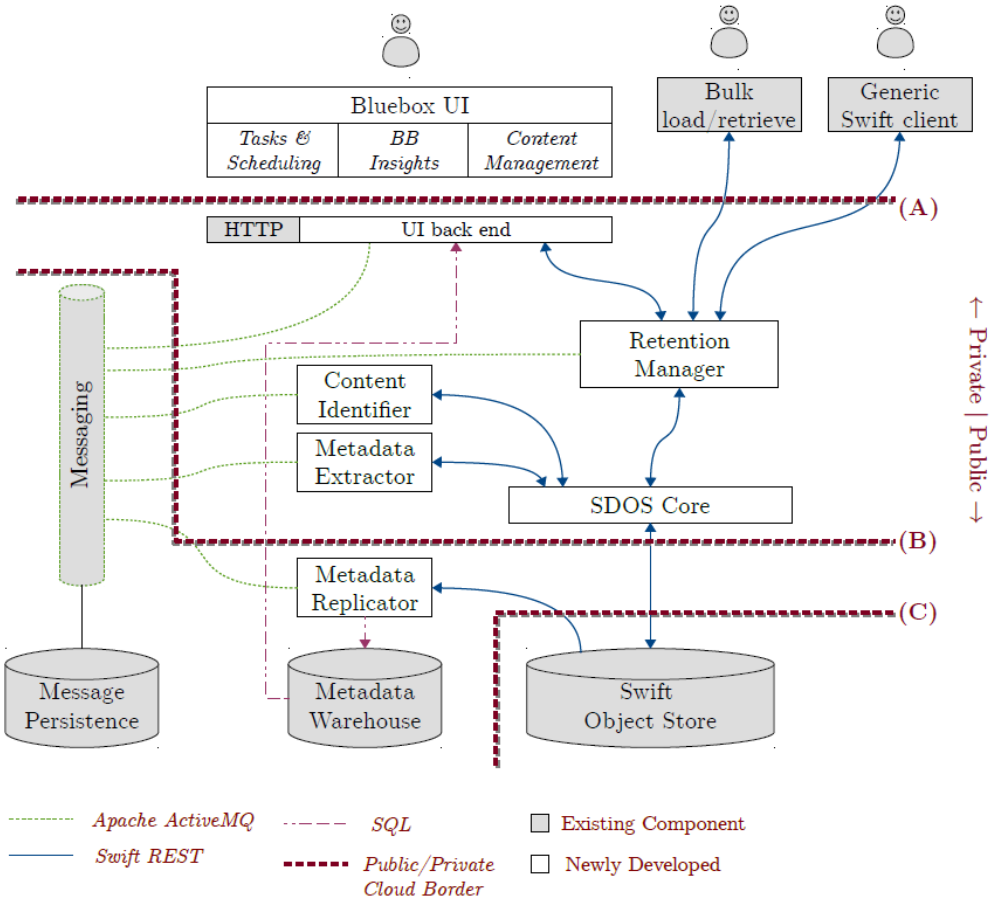

Figure 2.1: The OSECM Architecture [**OSECM**]

Although Swift supports both data and metadata natively and keeps some information like name, size and creation date of the object as a default set of metadata list for each object, sometimes more metadata can be extracted from each object to be used later on for advanced search and analysis. For this purpose, I create three components that are used to automatically create and fetch some metadata from objects, and store them inside Swift Object Store as well as replicate them to the metadata warehouse for further usage. The general extraction process is shown in the following picture:

In fact, in order to support more complicate and precise analysis scenarios, I fetch more metadata from each object based on its content type and store them inside Swift before retrieving for analytics. This way, Swift is kept as the central and main repository for both data and metadata and all changes have to be submitted inside Swift Object Store

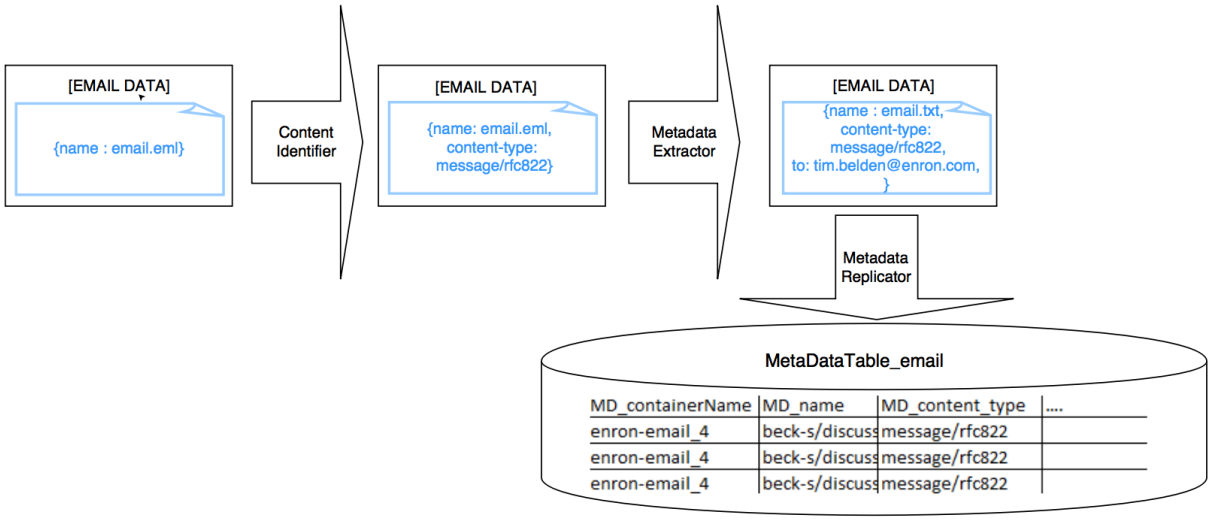

Figure 2.2: The process of extracting metadata from objects

to be used later on for analysis. Since Swift lacks advanced queries features as well as fetching and listing, in this work, I used a relational database as a replicated metadata warehouse to keep all the useful metadata inside one place. Furthermore, in this way, there is a unique metadata warehouse that is used as a single source for analytics and since it is based on SQL, all the SQL queries powers can be used to fetch the desired candidate data set for analysis use cases. Metadata replication in this system is only unidirectional and all changes of data and metadata occurs directly on the swift object store. This architecture helps to have only one main storage which should be taken into the account for all modifications and migration activities [**OSECM**].
Metadata which is the base for the analysis and could be used to support searching capabilities should be classified in a way that we can extract the desired information (e.g. fetching subject of an email or title of a PDF file) from them and replicate the extracted metadata into a metadata warehouse. For this purpose, as it is shown in the figure 2.2 as well, the following modules has been designed and implemented: Content Identifier, Metadata Extractor and Metadata Replicator. My contribution in the OSECM system is :

1. Providing the Metadata Extractor for email.
2. Designing and implementing the Metadata Replicator.
3. Designing the metadata warehouse schema.
4. Producing SQL queries for analysis use cases.
5. Defining the supported data structure for the visualization.

6. Preparing the appropriate set of visualization graphs using Bokeh library.

After metadata extraction process, we have a data management system which has a replica of metadata and can be used for queries and visualize the result. For instance, for email dataset, some metadata can be extracted in this way and then to be used for presenting a graph of people who communicate with each other. For this purpose, we need an analysis component that user can create its queries and apply them on metadata warehouse to get the result. Then I show the result to the user as plotted into the frame of different graphs based. As it is shown in the OSECM architecture picture, Bluebox UI is the user interface of this whole ECM system. Therefore I decided to show the result of analysis as a part of the Bluebox UI web application.

2.1 Swift Object Store

The Swift Object Store from the OpenStack project is an open source engine to store a big data (big in both contexts of number and size of the data) for cloud-based applications. It drives a major and large scale cloud service providers right now including companies like Rackspace, HP, Red Hat, IBM and some other famous enterprises. It is also used in quite a few private cloud deployment in order to store large amount of static contents that can grow without bounds. Swift is suitable for the data which is large scale, can require extremely high availability and demands high concurrency across the entire data set.

2.1.1 Swift Data Model

Swift has three types of entities that can be used for building the desired data model. These entities are from the type account, container and object. Basically an account is a user or an organization, who is the owner of the data. Each account can have multiple containers inside and each container can have several objects. Here in our system we have a flat data model for Swift object store hence we do not have sub-containers. To be able to implement a hierarchical file system structure with Swift, it is recommended to use the context of pseudo-paths[2] which means using the whole path to a file as the name for the swift object. This way the folder hierarchies are actually implemented in their names. Tracking objects can be done by prefix filtering which is supported by Swift.

[2] http://docs.openstack.org/user-guide/cli_swift_pseudo_hierarchical_folders_directories.html

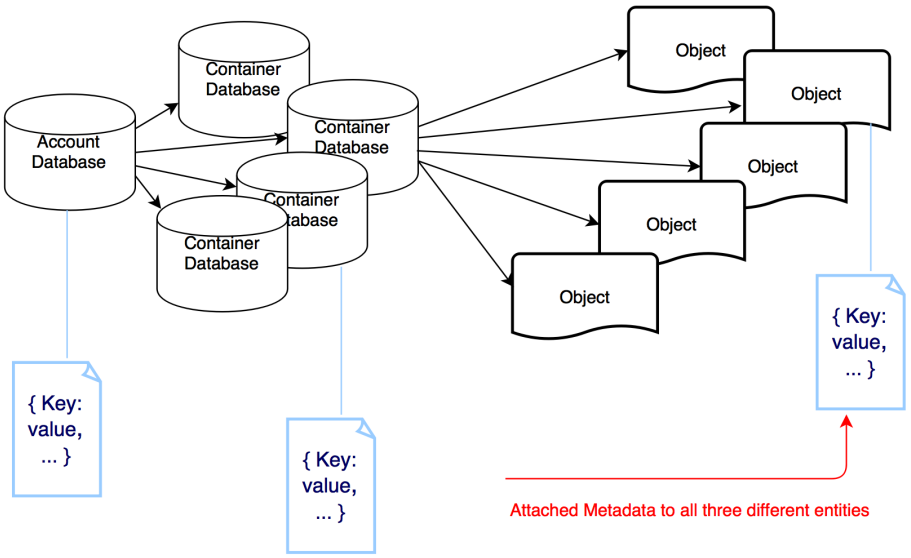

Figure 2.3: Swift Data Model; includes Accounts, Containers and Objects plus their attached metadata

2.1.2 Swift Metadata

Swift stores metadata for all its three various entities. These metadata are in a form of key-value pairs. These metadata are the information that Swift knows about the entity like name, size, date and etc. Besides, Swift can stores some extra defined metadata which are stored in entity itself and add them to the list of metadata for that entity. In this way we are using Swift as a central storage for our Enterprise Content Management system because we can store both data and metadata inside it. In our design this functionality is done by Metadata Extractor in our system.

2.2 Content Identifier

The content identifier distinguishes the type of an object through analyzing its content. Then it fills the "*content-type*" metadata field it with the determined type. The user also can set a content type for an object when uploads it into the Swift. But it has a problem that most of the users check file extension from the file system when uploading them as objects which are not reliable. Hence content identifier uses a specific programming library to distinguish the correct content type [**OSECM**].

2.3 Metadata Extractor

Since metadata is a critical part of each Enterprise Content Management system to find, categorize and organize data, having a larger and more precise collection of metadata can facilitate the ECM system profoundly. For this purpose, more metadata should be fetched from each object and this is done by Metadata Extractor in the OSECM system. After distinguishing the correct content type of objects, Metadata Extractor can use this information about the object structures and fetch more useful metadata fields from their structures.

2.4 Metadata Replicator

Swift Object Store lacks sophisticated querying capabilities for metadata as well as not being efficient for reading and listing of them. Therefore we decided first to replicate all the desired metadata to a relational metadata warehouse and then use this data management system to run queries.
The Metadata Replicator fetches the metadata from the Swift object store and put it into the database. The metadata comes from various sources:

1. Swift Object Store internal metadata,
2. Metadata retrieved by content type filters,
3. Metadata defined by user which belongs to a document class

Currently, the third group of the metadata is not implemented fully in our system but there is a possibility to add it to the system for future works. The replicator creates a table for each content type filter of metadata inside the database. These tables are joined together by referencing object and container name.

2.4.1 Internal Metadata

Swift creates some basic and default information as internal metadata for all different types of entities that it has. These information are usually name, size, type, access and the last-modified date and others. The replicator put all these information inside one table which is called "*MetaDataTable_SwiftInternal*" with the primary key of the combination of container name and object name.

2.4.2 Filter Metadata

Since the replicator has the same filter plug-in interface as the extractor, it can pick up metadata fields from the filter plug-ins. The replicator creates a table for each content type filter and choose the desired set of metadata fields from the object itself. It is very important to use the same version of the plug-in for both extractor and replicator in order to fetch and store the correct set of metadata.

2.4.3 Document Class Metadata

This group of metadata helps to categorize each types of entities to different classes as user defines them. It is managed similar to filter metadata. The replicator creates a table for each document class with the required columns based on its definition. The document class definition should be extracted from the Swift. At the moment, document class is only applied on the containers which means every object inside this container refers to that document class. Therefore, all the specified metadata fields from that document class can be read from the replicator and stored into the database. Users can define some other types of metadata that are not belong to any of the filter, document or internal metadata groups and as Waizenegger et al. mentioned, they are called "free" metadata [**OSECM**]. Currently our system ignores the free metadata since the relevant schema is not known for them.

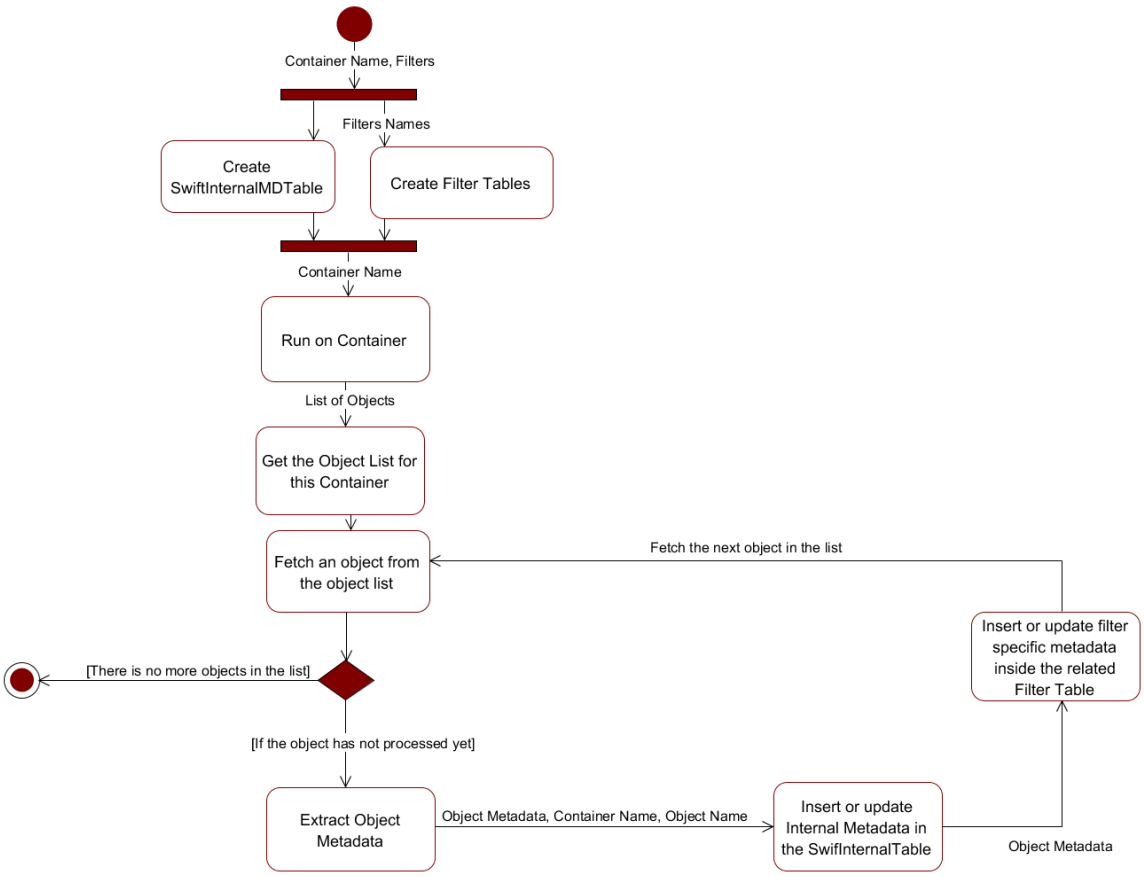

Figure 2.4: The Metadata Replication Workflow

3 Analysis Concepts

Analysis refers to breaking a whole into its separate components for individual examination. In fact, data analysis is a process of fetching raw data and transforming it into information useful for further usages like supporting decision making systems and offering sophisticated . Data is collected and analyzed to answer questions, test hypotheses or disprove theories. Here, in order to provide analysis features for the OSECM system, all the available data has to be collected, cleaned and then fetched to the data management system to be ready for analysis queries. The analysis application runs analysis queries and send its results to the visualization system Which is Blebox UI system for the whole OSECM system. Then, the visualization system should use a set of standard graphs and diagrams to serve the result for the user in order to be able to find out more about the hidden insights of the data. Generally, the figure 3.1 shows the required components and architecture that such an analysis layer should have using the Swift object store as the back end repository.

The analysis application connects to the metadata warehouse which contains all the available and useful metadata which are extracted from Swift object store. This application should provide a possibility for users to define analysis queries in a flexible manner. The possibility to define queries should be in a way that the implementation of the analysis application has not been changed to add a new query or modifying the existed ones. In this architecture, a relational data management system is used in order to provide a comprehensive support of SQL queries and the possibility to actually run the queries inside the database to return the proper result. Since the provided results have various structures, the visualization system has to support a collection of defined result data sets. Finally a suitable graph also should be chosen to present the final result.

To support the presented architecture for this analysis layer, different options are possible:
First option would be having a completely new analysis component, consists of the target queries that should connect to the metadata warehouse and get the result back for the visualization. But in this case, for designing each new analysis use case, the application should be changed and new queries should be added to the system which does not bring much flexibility to the system. Furthermore in this case, the new component should be created from the scratch and in the case that we would be restricted to use a specific database or data warehouse as our metadata warehouse, this application also should be

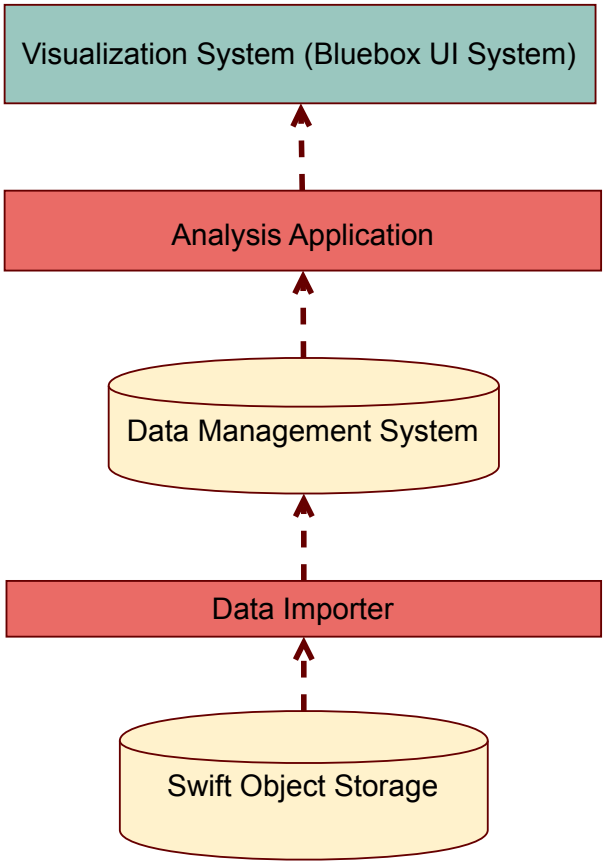

Figure 3.1: The required components and architecture to create the analysis layer; Considering that object store and Bluebox UI is existed

changed in order to support both connection to the new database and the new syntax of the used database to write the queries which could be supported by this database. Another option is using an existed analysis open source tool and integrate it with the OS-ECM system. Here we decided to use Node-RED[1] tool which is basically a visual tool to make Internet of things applications. It provides a browser-based flow editor that makes it easy to wire together flows using the wide range nodes in the palette. Hence, user can define all the required steps including get the request, fetch the data from the metadata warehouse, write desired queries, run them inside the database, get the result back, and create an endpoint to make the result available for the external usages like plotting data for visualization purposes. Moreover, the additional processing of the result is possible through defining JavaScript functions and retrieve data from other external sources.

[1] http://nodered.org/

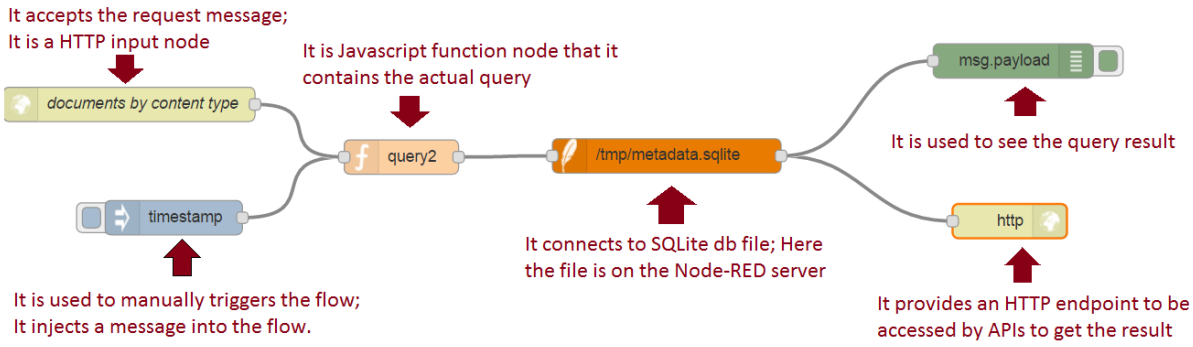

Figure 3.2: The analysis scenarios inside Node-RED Server; providing an Endpoint for outside access

In this system, I design the flows inside Node-RED server in a way that each of them should have an HTTP input node to get the HTTP request coming from the end user side and an HTTP output node to create an endpoint for presenting the result for the visualization application. In this work, in order to make provide the analytics feature accessibility by the end user, we create a new part in the Bluebox UI system which allows the user to open Node-RED environment to define a new analysis flow that could be reused, choose the newly created or existed HTTP endpoints inside the Node-RED server to get the desire analysis result set and choose the proper visualization graph to see the result.

3.1 Visualization

Data visualization mainly deals with the techniques used to communicate data or information by transforming it as visual objects (e.g., points, lines or bars) existed in graphics. The target is to communicate information apparently and accurately to users. As I mentioned in the background chapter, the visualization is one of the main steps in data analysis or data science. According to Friedman the "main goal of data visualization is to communicate information clearly and effectively through graphical means [Fri08]. The data visualization does not have to be looked tedious to be functional or highly complex to look pretty. In order to exchange ideas and information efficiently, both inventive form and good functionality need to be taken to the account. In this way, insights can be delivered into a rather sparse and complex data set by communicating

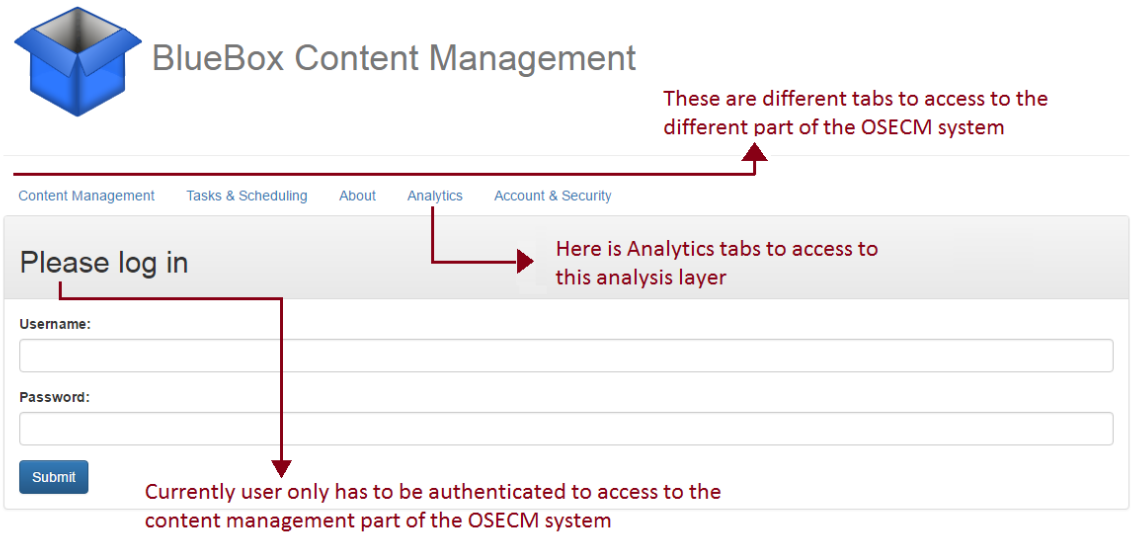

Figure 3.3: Different part of the Bluebox UI

its main aspects in a more perceptual way.
Here, in order to meet both the main targets of visualization including being aesthetic enough and conveying useful information too, to create analytics diagrams, I have used Bokeh[2] which is a Python interactive visualization library that targets modern web browsers for presentation. Moreover, for some of the implemented analytics use cases, Numpy[3] and Pandas[4] besides Bokeh to provide other aspects like managing the data frame.

3.1.1 Bluebox UI

End user can access to the analysis feature of our ECM system through Bluebox UI. Generally communication with the whole OSECM system is possible both through Bluebox UI and Swift API.
In order to perform different functionalities of the existed version of OSECM system, currently Blubox UI has 4 parts as it is shown in the figure below. These parts are designed as different tabs in one page application using AngularJS framework and Bootsrap library to be integrated with the Bluebox UI.

[2] http://bokeh.pydata.org/en/latest/docs/dev_guide/
[3] http://www.numpy.org/
[4] http://pandas.pydata.org/

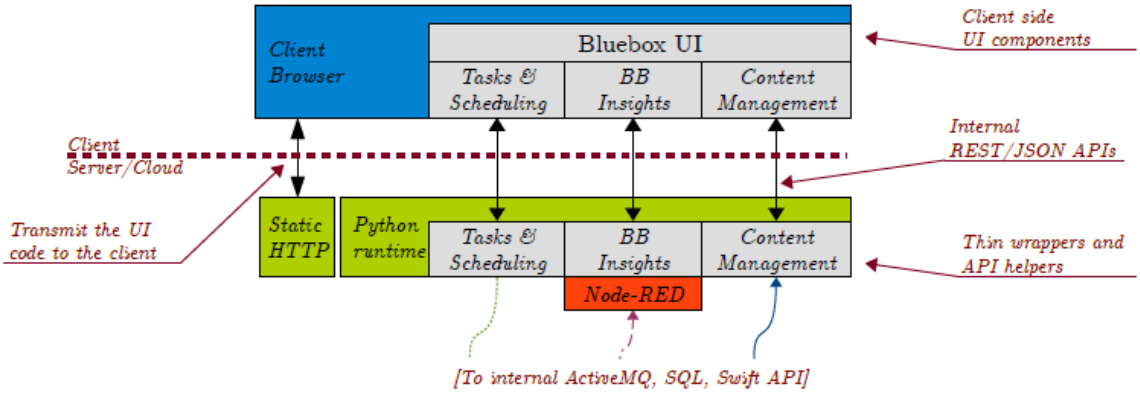

Figure 3.4: The Component Overview of Bluebox User Interface [**OSECM**]

In fact Bluebox UI has exactly three functional components that are linked to the three internal APIs which are used:

As it is shown in the figure 3.4 , in the following, I describe the different parts of the Bluebox UI system based on the OSECM paper [**OSECM**].

- Tasks and Scheduling
- BB Insights
- Content Management

Tasks and Scheduling is connected to the internal Apache ActiveMQ messaging service. Users can perform tasks triggering, scheduling besides getting responses from the tasks. At this moment these tasks are used for executing content identification, metadata extraction, and metadata replication.
BB Insights includes the analytics and statistics component. It works with a Node-RED[5] instance that is connected to the metadata warehouse. This way, user can create SQL queries and connect to the external data sources which is in this work SQLite database file. Then user can proceed with using JavaScript functions to analyze the desired data and prepare it for the visualization. Entirely it accomplishes the following operations: First, it presents the available schemas and data in the metadata warehouse for the user. Then, it allows accessing a Node-RED instance which has a SQL connection to the metadata warehouse. Currently in this prototype, for this purpose SQLite database

[5] http://nodered.org/

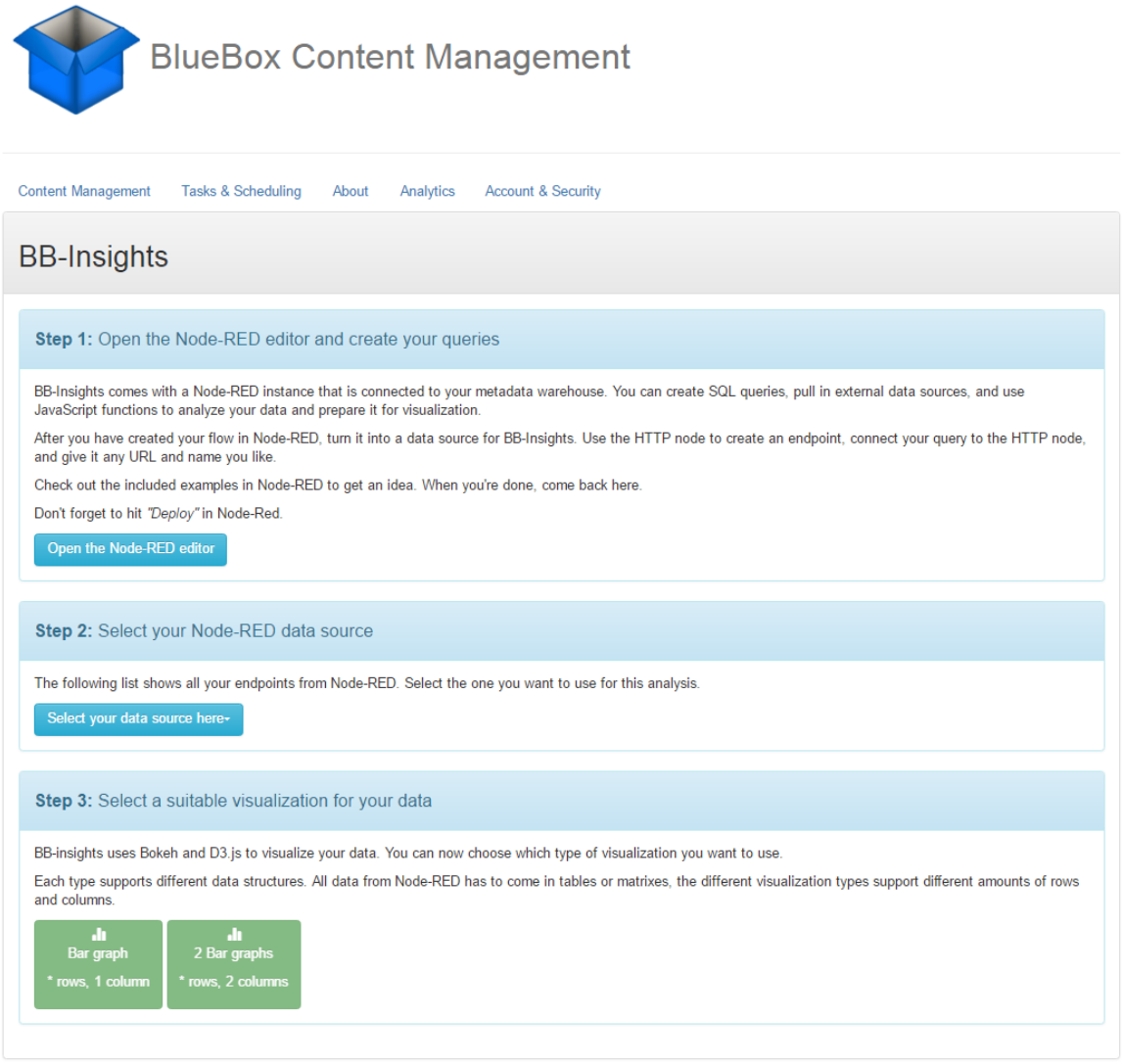

Figure 3.5: The BB-Insights User Interface inside Bluebox

file is put in the Node-RED server. Hence user can access to all data that replicated to the metadata warehouse and create SQL queries for the targeted analysis scenarios. Similarly user can graphically link all the SQL queries together with JavaScript functions inside Node-RED. Finally, it provides endpoints for Node-RED to output the analysis results. This data is then delivered to the client application by the API wrapper on the server side. The server application then locally renders the data into graphs in the clients web browser.

Content Management is the other UI component in Bluebox. User can communicate with the Swift object store to insert and fetch the data. A file browser is provided like interface to the object store with capabilities to manage metadata, containers and the SDOS functions.

3.1.2 Analytics Diagrams

In order to support a collection of result data structure that can be visualized in this system, I defined a group of data structure that can be plotted by a specific set of graph types as well. As it is shown in the figure 3.6, I define three types of data structure that the result should be looked like.

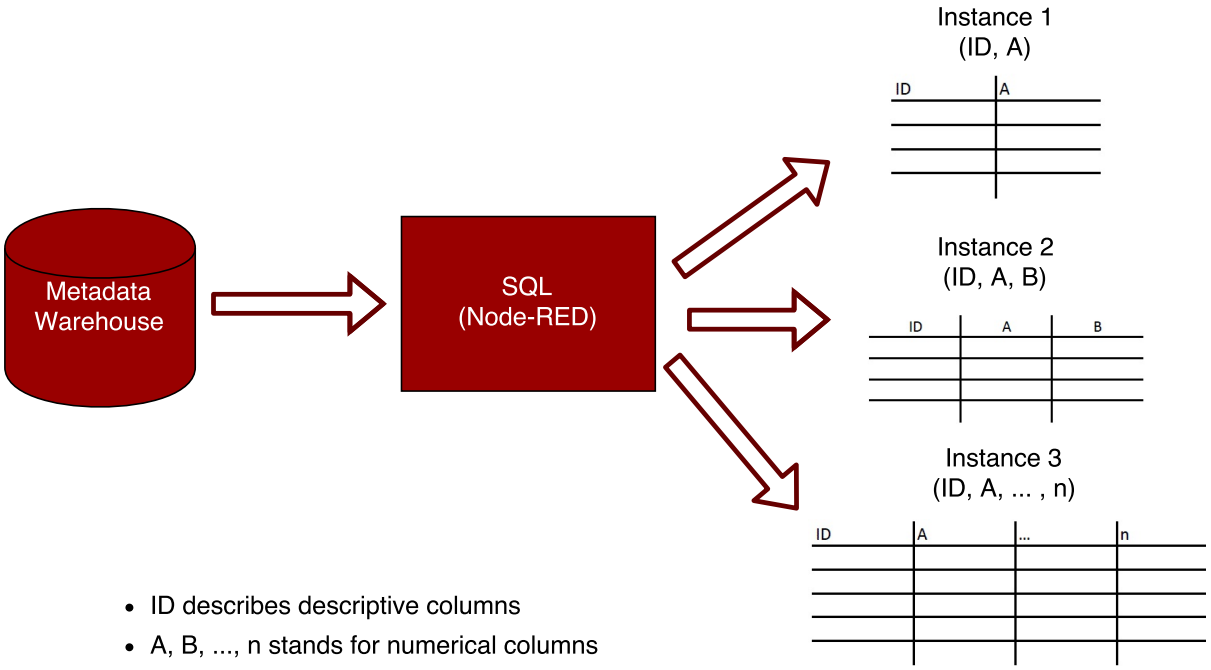

Figure 3.6: The supported data structure for the visualization

For each instance of data structure, there is a set of defined graph types, that the result can be shown with them. I describe the list of suitable graph that can be used for each of the data structure types:

Instance One The data structure has exactly one numerical column and multiple rows. The rows are identified by an id column. For this type, in our system I support **Bar Graph**.

Instance Two The data structure has exactly two numerical columns and multiple rows. Each row is identified by an id column. The numerical columns are plotted against each other through using **Line Graph** and **Area Graph**.

Instance Three The data structure has two or more numerical columns beside multiple rows. Each row is identified by an id column. For this type, currently this system supports stacked bar graph, box plot and donut chart.

In the Section 4.7, the definition of available graphs and some examples from the OSECM system are presented.

3.2 Candidate Use cases

1. Showing the number of objects for different types of containers.
2. How many containers/objects does each owner have.
3. How many containers/objects are expired.
4. List of objects which are recently modified (Within 0-10 last days).
5. Ranking containers based on their number of objects.
6. Recognizing objects as an image and put them inside image container.
7. Recognizing the network of people who are interacting with each other through Emails.
8. Identifying which period of time has the large amount of sent Emails
9. Using text analysis and natural language processing to search some specific clause in the email.
10. response time
11. grouping people, who postpone whom? is it depend on their position in the organization structure (Are they closed colleagues or not.)

12. Formal or informal sent email based on group of people like working group or age group.
13. Which subjects are the main subjects
14. Email frequency or volume of emails between different group of people
15. Each person has some properties like department, experience years, etc.
16. How effective emails are in working subjects.
17. How much of people in cc and bcc have any reflection in order to response to their emails. For instance recognizing manager who has not had any reflection to their emails.
18. Calculating the Response time for each email.
19. Tagging people based on organization structure.
20. The influence of position, age, type of work, ... on the response time.
21. The influence of people position on the read or unread emails.
22. How much people in different positions are in cc and bcc groups.
23. To find out what the position of the people is by looking at the data. For instance based on some structure that there existed inside emails you can find out who is this person? Is he developer or HR employee or boss or whom? Based on the way that they communicate and so on.

4 Technology and Implementation

For all components that are described in the architecture chapter and involved in the analysis layer, different technologies has been employed to create new components and integrate them with the OSECM system. For some of the used technologies, I tried other alternatives (e.g. to make visualization) as well but at the end I tried to choose the most appropriate ones to make the whole prototype flexible enough. For instance, for the visualization part, to make the graphs for final result, using D3.js also is tried but since making interactive graphs with D3.js need more coding and it is more complicated, we decided to use Bokeh[1] library in cooperation with other open source libraries like numpy [2] and pandas [3] to make the appropriate graphs.

In this chapter, for each part of the system, I describe the tools and technologies that are used as well as explaining the way that I implement the required components and integrate them with the OSECM system.

4.1 How does Swift work?

Swift has two parts: a proxy server and storage nodes. On the front end there is a proxy server which accepts all the requests from end users and is responsible to implement all the APIs that swift provides. The proxy server controls all communications with back end systems should be done correctly.[4]
This proxy server communicates with storage nodes and ensure that clients get their responses back. All the swift APIs are standard HTTP based API. When for instance client have a PUT request, the request goes to the proxy server and proxy communicates with the proper storage node and ensure that data is getting written down inside storage node. A storage node itself is responsible to durably store data inside disk and then sending the response back to the proxy server. Swift is replicated based system which

[1] http://bokeh.pydata.org/en/latest/
[2] http://www.numpy.org/
[3] http://pandas.pydata.org/
[4] Swift Online Content: https://www.youtube.com/watch?v=z0jQFoW_QgM

means all the data inside it will be stored multiple times to ensure high availability and durability. It is recommended to use 3 replicas by default which means use three storage nodes. Storage nodes are communicating with each other to ensure that data is correct and is in the right place.

How swift choose the place to store the data? Swift uses a concept that is called a "Ring" to place the data. When request comes in, it first looks which appropriate storage nodes is responsible to store this data. This nodes are getting choosed such that they are in isolated value domains. It organizes places based on the following order of categories: 1- region , 2- zone, 3- server, 4- drive. Since swift supports global clustering, it is possible to have globally distributed clusters which are treated as one logical swift point. It means you can have deployments on east coast and west coast and swift treats it as one point which manages by means of "Region" and gives you a good durability and support of disaster recovery. This makes better availability as well because if you have users who access the data from different locations, they can preferably goes to one of nodes which is close to them. This makes higher throughput because you can have lower network latency.

4.1.1 Swift REST API

We use Swift Rest API to communicate with Swift itself and for communication between other modules as well. In addition, this API is used for user interface too. Since the Swift API is universal and it is based on the Rest architecture, it can support all the main functionalities to create, read, update and delete (CRUD) the entities. Furthermore in our ECM system, we are using the same data model as Swift to manage the contents. Therefore we develop our components based on Swift API since our data structure is as same as Swift API. This way we can integrate any other applications (e.g. analysis applications) that can work with Swift back end system, with our ECM system which brings more flexibility to us. As it is mentioned in OSECM paper [**OSECM**], another advantage of using tis API would be a possibility to create a swift client or a Swift object store with a collection of components.

4.2 Content Identifier

Content identifier uses the programming library magic from the well known Unix file[5] command. It includes a complete list of common content types and their binary fingerprints. In fact, this component checks the binary content of the object to find certain sequence of bytes that determines the content type.

4.3 Metadata Extractor

Metadata Extractor extracts more metadata for each object based on its content type. In fact, it provides an execution environment and plug-in interface for content type filters. The supported content types and list of desired metadata have to be specified for all filter plug-ins. First the extractor, based on the content type field, distinguishes the related plug-in for each object. Then it downloads the binary content of the object, and sends this data object to the filter plug-in. The found data is returned as key/value pairs by plug-ins and then the extractor updates object metadata inside object store with the new extracted list. Currently there are different filters for the following object types which are existed previously: Images with JPEG, PNG, BMP, ... formats, PDF documents and I added an email filter during the course of this work. For example, the email filter may provide metadata fields like "to", "from" , "subject", "date" and others or the PDF filter may provide "author" or "title". In the OSECM system, to extract metadata for each of the supported formats, specific libraries are used. Since the filter plug-ins defines a list of metadata fields that they may produce and the related content type for this filter, this information gets used in the replicator component later to generate a database schema for each content type, in the analytics component to suggest fields for analysis, and in the user interface to display column names to the user [**OSECM**]. In the figure 4.1, the Metadata Extractor class diagram is pictured.

4.4 Metadata Replicator

As the course of this wrok, I implemented the Metadata Replicator in order to publish the extracted metadata into the metadata warehouse. The Metadata Replicator uses the same interface as Metadata Extractor for filter plug-ins. Therefore, the Metadata Replicator first creates one table to keep all the general information of the objects namely

[5] https://github.com/file/file

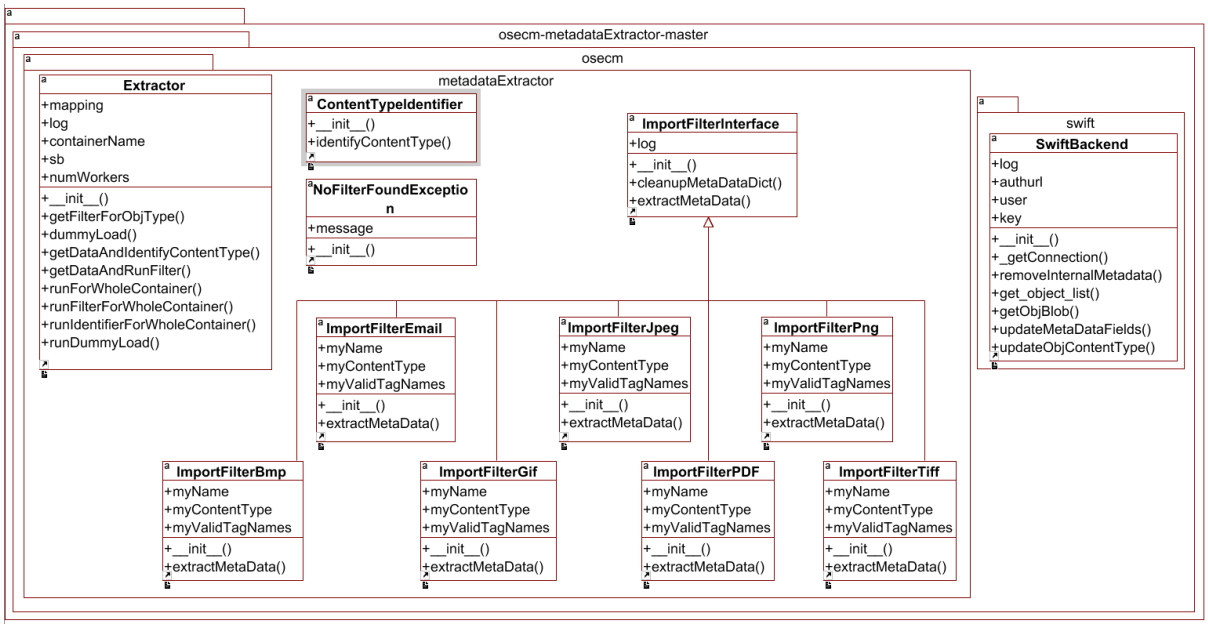

Figure 4.1: The Metadata Extractor Diagram

by object name, container name, content type, content size and others. Then, it creates filter tables and for each filter table, it uses the list of metadata in the filter table as the columns. After that, it gets the object list for this container. For each object, it fetches both the internal and filter metadata and first insert the internal ones into the general table and then insert the filter metadata into the appropriate filter table. To make a relation between these tables, the combination of the object name and container name is used as the primary key in the main table *("MetaDataTable_SwiftInternal")* and as the foreign key in the filter tables. The figure 4.2 shows the class diagram of the Metadata Replicator.

4.5 DataModel

This system keeps the data for each object within two levels of tables. First, we have a main table which is called "*MetaDataTable_SwiftInternal*" which keeps all the useful internal metadata plus the date and time of metadata extraction from the object store (that can be used to monitor how much the data inside this table is updated). This table has a primary key which is the combination of container and object name. Then, for each content type filter that now this system supports there is a table with the appropriate collection of columns which are extracted from the metadata set of each object inside object store. Currently the available filter types include: email format, pdf files and

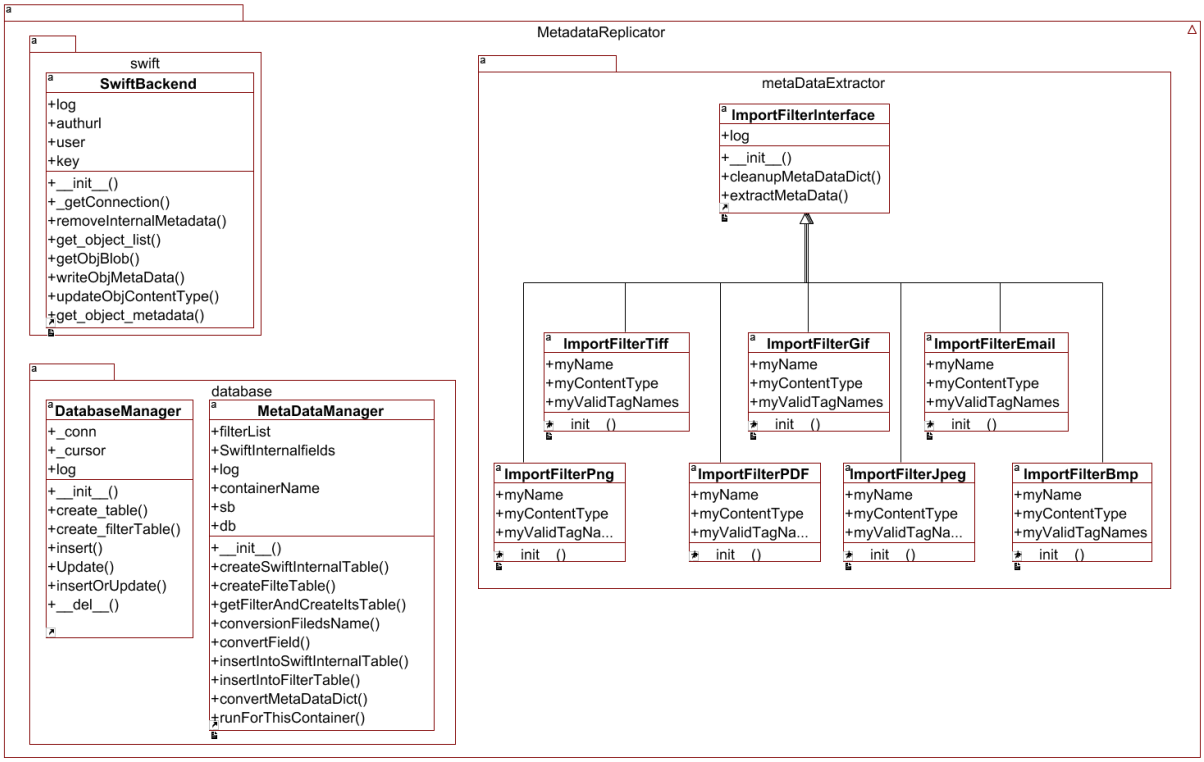

Figure 4.2: The Metadata Replicator Class Diagram

some image formats consist of tiff, png, jpeg, gif, bmp.
For each object first we extract a fixed set of metadata that are defined as the columns for the main table *("MetaDataTable_SwiftInternal")*. After that, based on the content-type of the object , a related collection of metadata gets extracted to be put in the allocated metadata filter table. For example, for each email object, first a record is filled inside the main table, then to store further metadata (e.g. to whom this email is sent, who was is the "cc" field for this email and others) which are specifically related to the email objects, a row inside "*MetaDataTable_email*" should be created. With this data structure, writing the analytics queries is easier. To keep the connection between these two level of tables, the collection of object name and name of the container which this object is stored there, is used as foreign key inside filter table to refer to the main table. In the following the whole data model is shown:

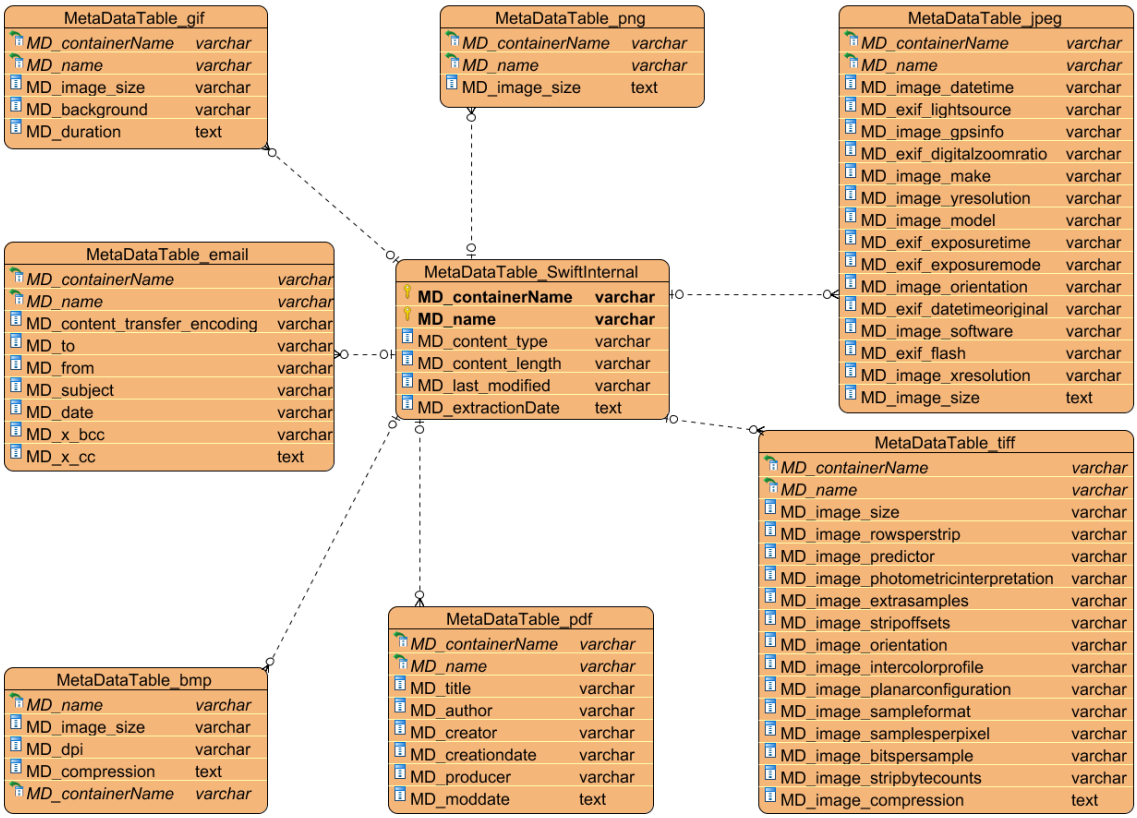

Figure 4.3: The Metadata warehouse schema in SQLite database

4.6 Bluebox UI

Bluebox UI has two layers. The first layer on top consists of the client side UI components that run as an AngularJS[6] application inside the client's browser. The second layer on bottom is on the server side which includes a static HTTP server that sends the client application to the browser and a Python runtime that provides a back end for this application. The back end side produces data in JSON format via a REST based API for the client side using a group of small API wrappers to communicate with SQLite, Swift and ActiveMQ (which is used for other parts of the ECM system no related to the analysis) and Node-RED. This back end does not keep any application or session status so that it can scale horizontally as well as HTTP server because it only provides static content to the client [**OSECM**].

[6] https://angularjs.org/

4.7 Analytics Diagrams

To create the analytics diagrams, I used Bokeh[7] python library that it brings some advantages for the visualization in this analysis system as it is an **interactive** visualization library for **python**. In fact, Bokeh is created to extend the novel graphics nature of D3.js[8] (which is another powerful visualization framework to create modern web browsers) capability with high-performance interactivity over very large or streaming datasets. It is designed in a way that program does not have to be written in JavaScript and can be implemented as a standalone or server-based web application.
In the OSECM system, in order to use the **Analytics** component, first user can open the Node-RED environment to design a new analysis flow. In case that it wants to use a result data set of the existed analysis flow, it can access to the list of current result data set and select the desired one. Then it should select the visualization type which distinguishes a convenient diagram to plot the result. These information is sent back to the Bluebox analytics (BBInsights) server which will then get the respective data set from the Node-RED and run through the Bokeh in order to provide a piece of JavaScript code that it pushes back to the client.
The client, which already has the Bokeh library, needs the data and commands to draw the result. Hence, the required data is given to the client by the JavaScript code which is produced by the Bokeh inside the BB Insights server component. In fact, BB Insights component of Bluebox UI system, has a small part on the server, which runs Bokeh. This part provides the plot data for the user interface which will be integrated to the existed Bluebox UI system.

For most use cases in this work, I used bokeh.charts[9] and bokeh.plotting[10] interfaces to plot the data. Usually for stand alone bokeh applications like my system, when using the bokeh.plotting or bokeh.charts interfaces, users call the function output-file() in conjunction with show() or save() instead. Here, in order to be integrated with the AngularJS framework which is used for the whole Bluebox UI system, we use function **components()** which returns HTML components to embed a Bokeh plot. The data for plotting in this case is stored directly in the returned HTML.The components()[11] function takes either a single Bokeh Model, a list/tuple of Models, or a dictionary of keys and Models. In this way, in OSECM system, it can support creating both single or collection of diagrams to present the plot data.

[7] http://bokeh.pydata.org/
[8] https://d3js.org/
[9] http://bokeh.pydata.org/en/0.11.0/docs/reference/charts.html#bokeh-charts
[10] http://bokeh.pydata.org/en/0.11.0/docs/reference/plotting.html#bokeh-plotting
[11] http://bokeh.pydata.org/en/0.11.0/docs/user_guide/embed.html

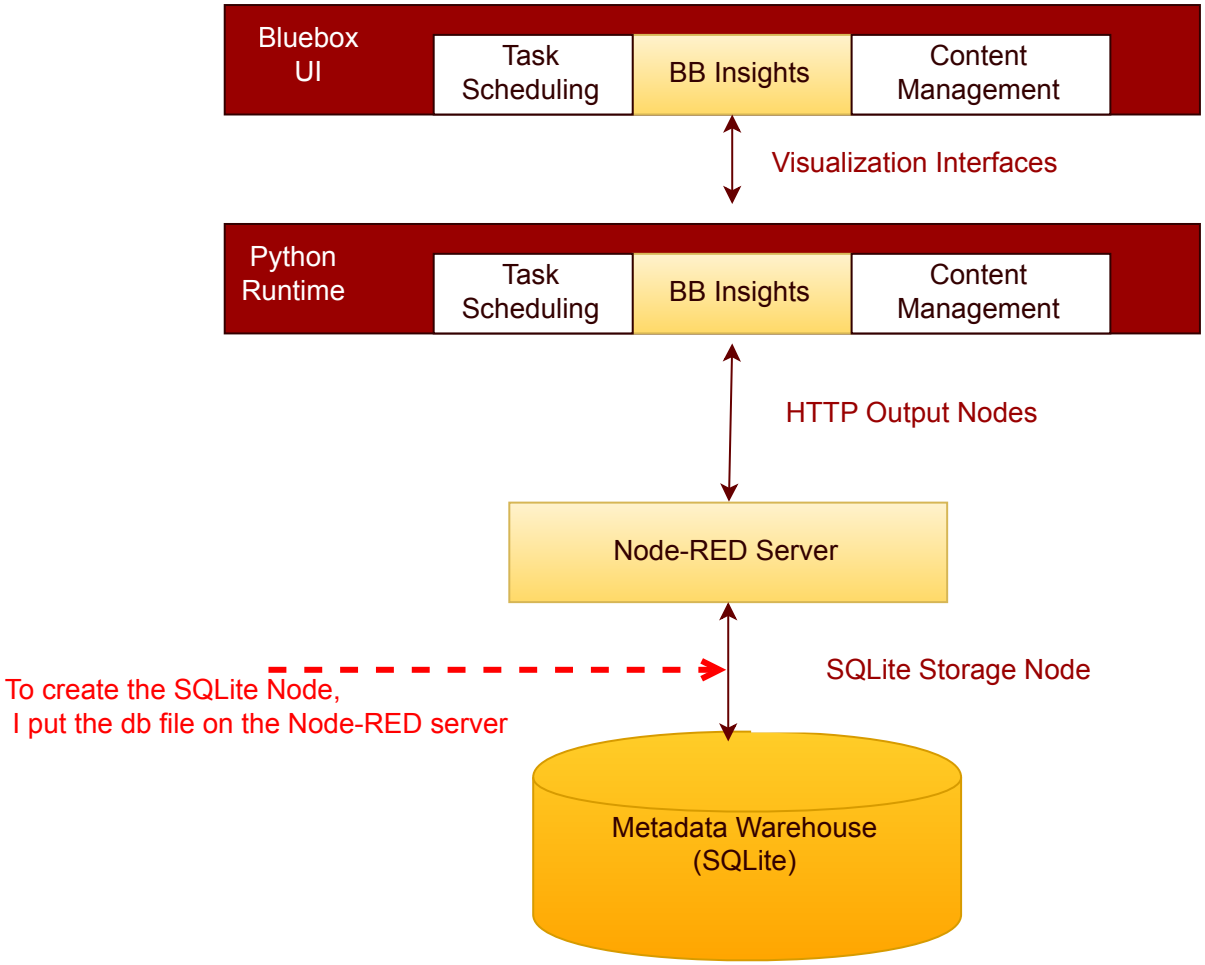

Figure 4.4: The Analytics Component Interactions

- Single Graph For this case, currently our system supports **Bar Graphs**[12] and **Line Graphs**[13]. These graphs are used for the case that the target data set has one quantitative column and the distribution of this column is considered to be plotted. The graphs which are shown in figures **??** and 4.7, as a bar graph and line graph respectively.
- Multiple Graphs When the analysis result set which is coming from an API endpoint of the Node-RED server, has more than one quantitative column, the OSECM shows all the possible graphs for this data set through making multiple graphs with the appropriate graph type based on the data set structure. For instance, The OSECM

[12] https://en.wikipedia.org/wiki/Bar_chart
[13] https://en.wikipedia.org/wiki/Line_graph

select MD_content_type, count(MD_content_type) as 'number of docs', sum(MD_content_length)/1024 as 'size of docs in KB', sum(MD_content_length) as 'size of docs in bytes' from MetaDataTable_SwiftInternal GROUP BY MD_content_type;"

Figure 4.5: The Content Type Numbers and Size Query

shows three graphs with the same types (e.g. all three are bar graphs or line graphs) for the result set of the analysis query which is shown in figure 4.5. This option is available through choosing **Multiple Bar Graphs** button in the Bluebox UI, Analytics tab. However, such a data set with has more than one quantitative column, can be plotted into some other types of graphs which even can compare these columns with each other in one graph. In this work I support of these graph types like stack or group bar graphs, box plot, donut chart and others. Later in this chapter, I explain them in more details.

In the following I show some examples of visualization graphs which can be support a set of data set with the defined characteristics. The BB Insights component then exposes the plot data, using either a simple or a complex Bokeh diagram which is compatible with the desired data set to be shown.

4.7.1 Bar Graph

Bar graph can normally be used to show the relative sizes of many attributes. In this case, the data set to be plotted has to have at least one numerical column and can have arbitrary number of rows. The data set can have one descriptive or id column as well. For instance, in the following the sample data set, the query to fetch the data set and plot data into the Bar graph is shown.

Listing 4.1 The Content Type Distribution Query

```
select MD_content_type, count(MD_content_type) as 'number of docs' from
    MetaDataTable_SwiftInternal GROUP BY MD_content_type
```

4.7.2 Line Graph

Line graphs are another basic charts that is supported by Bokeh library. The data set that can be plotted by line graph has to have two numerical and quantitative columns. One of these columns should be defined as an index column. It has to have unique and

Table 4.1: The content type distribution data set

MD_content_type	number of docs
application/CDFV2	2
application/CDFV2-encrypted	1
application/gzip	4
application/octet-stream	255
application/pdf	3537
application/vnd.ms-fontobject	1
application/vnd.ms-opentype	43
application/x-bzip2	1
...	...

continuous values which can be ordered if needed. In the following, I show an example of the object distributions based on their size.

Listing 4.2 The query to expose the object distribution based on their size

```
SELECT ROUND(MD_content_length/100000, 0) as "100KBs", count(MD_content_length) as
    "number of objects" FROM MetaDataTable_SwiftInternal GROUP BY
    ROUND(MD_content_length/100000, 0) ORDER BY ROUND(MD_content_length/100000, 0)
```

4.7.3 Area Graph

Area graph is a line graph with the area below the line filled in with a certain color or texture. Similar to line graphs, area graphs are used to display the development of quantitative values over an interval or time period. They are most commonly used to show trends and relationships, rather then convey specific values[14]. To support this diagram in this system, the data set should have at least two numerical and order able columns that can be plot against each other, same as a line graph. The first column is used as an index for x_axis which should have a continuous value and the second one would be plot against this one. In case that there are more than two numerical columns, stack area graph will expose the result set. The query in figure **??** is one sample query to use the data sets inside the OSECM system.

[14] http://www.datavizcatalogue.com/methods/area_graph.html

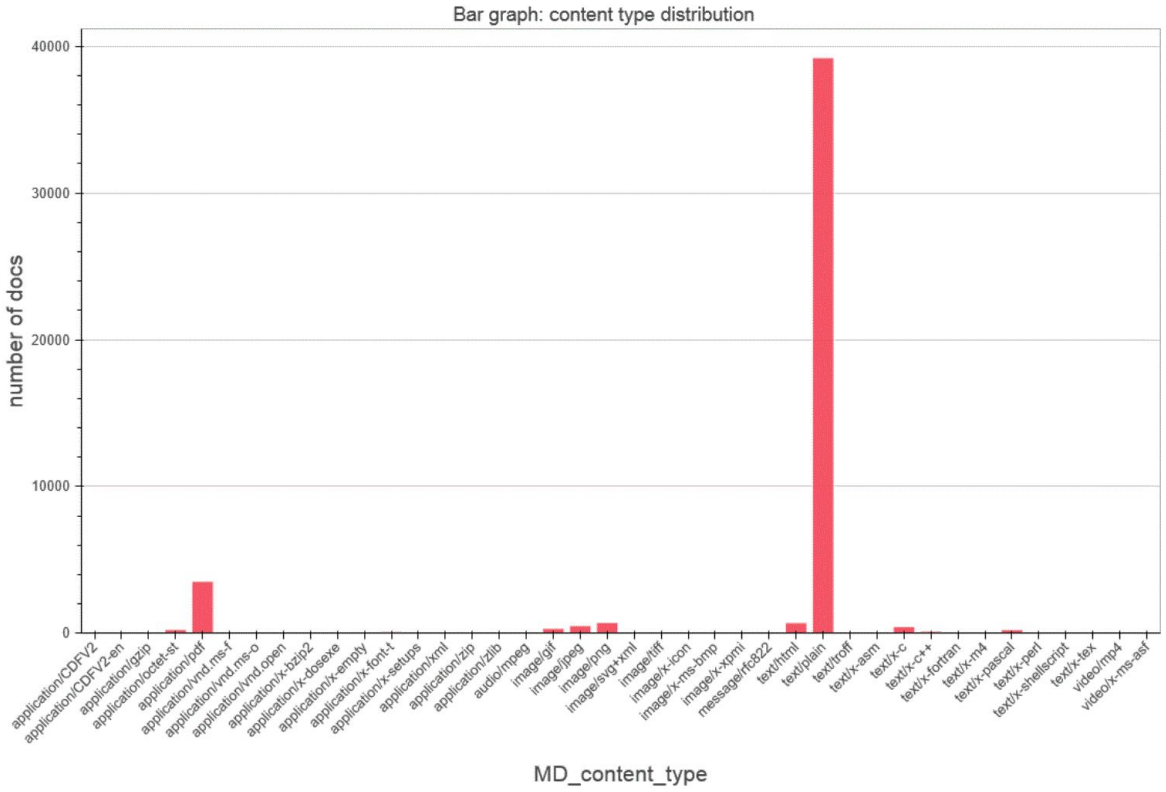

Figure 4.6: The Content Type Distribution Bar Graph

Listing 4.3 The Content Type Numbers and Size Query

```
SELECT ROUND(MD_content_length/100000, 0) as '100KBs', count(MD_content_length) as
    'number of objects', count(MD_content_length)+100 as 'number of objects2' FROM
    MetaDataTable_SwiftInternal GROUP BY ROUND(MD_content_length/100000, 0) ORDER BY
    ROUND(MD_content_length/100000, 0);
```

4.7.4 Stacked Bar Graph

Stacked bar graph depicts items stacked one on top (column) of the other or side-by-side (bar), differentiated by colored bars or strips. It integrates different data sets to create a richer picture of (the sum of) changes. A stacked Bar graph is useful for looking at changes in, for example, in this work the number of objects per its content type, comparing with the size of those objects in megabytes[15]. In the figure 4.9, the result stacked bar chart of this example is shown.

[15] http://peltiertech.com/stacked-bar-chart-alternatives/

Table 4.2: The data set of the object distributions based on their size. It is the result for the query of figure **??**.

100KBs	number of objects
0	42889
1	520
2	441
3	489
4	360
5	270
6	254
7	205
8	171
9	123
10	97
11	91
12	69
13	61
14	47
15	41
16	28
17	35
18	29
...	...

There is a possibility to visualize the groups in the data into **Group Bar Graph** using the group parameter in the implementation. In the figure 4.10 , the result of the mentioned example is shown in the group bar graph.

4.7.5 Box Plot

Box Plots can be used to summarize the statistical properties of different groups of data. It is a convenient way of graphically depicting groups of numerical data through their quartiles[16]. The label specifies a column in the data to group by, and a box plot is generated for each group. In the OSECM system, the data structure which be plotted

[16] http://stat.ethz.ch/R-manual/R-devel/library/grDevices/html/boxplot.stats.html

into a box plot has to have at least one numerical column to show its distribution for each group. This column defines the value set for each group. There can be some descriptive columns to define a group based of them. Here, as an example I pictured the result set that has two numerical columns. The query is mentioned in the figure **??** into a box plot.

Listing 4.4 The query to fetch the distribution of size and number of documents for each content type.

```
select MD_content_type, count(MD_content_type) as 'number of docs',
    sum(MD_content_length)/1024 as 'size of docs in KB' from MetaDataTable_SwiftInternal
    GROUP BY MD_content_type order by 'size of docs in KB' desc limit 20 offset 15;
```

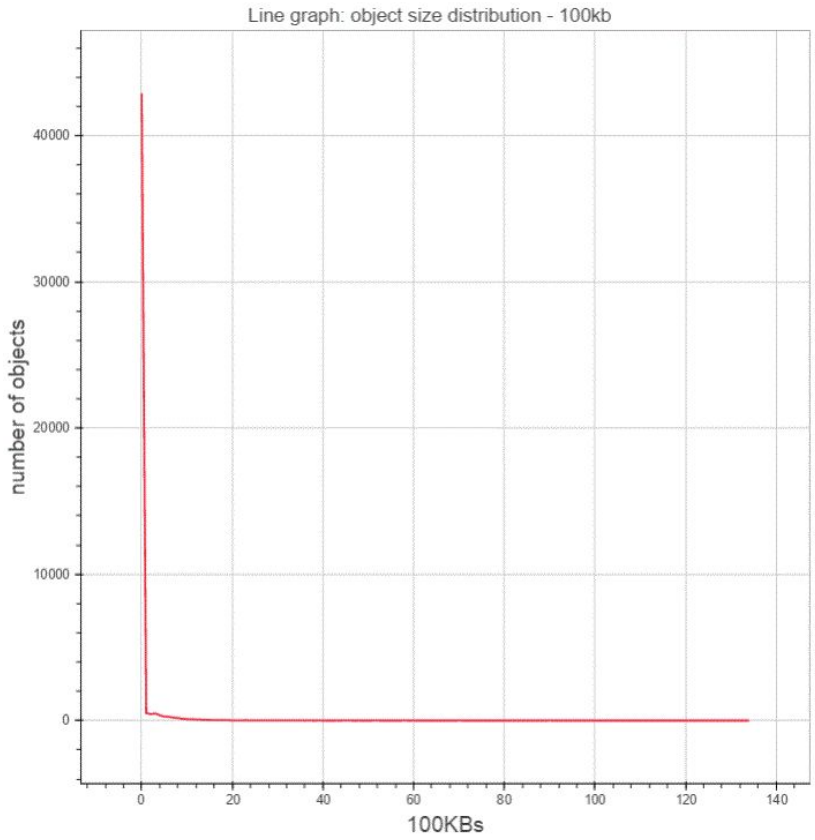

Figure 4.7: The Line Graph of the Object Distributions based on their Size; The query is mentioned in figure **??**

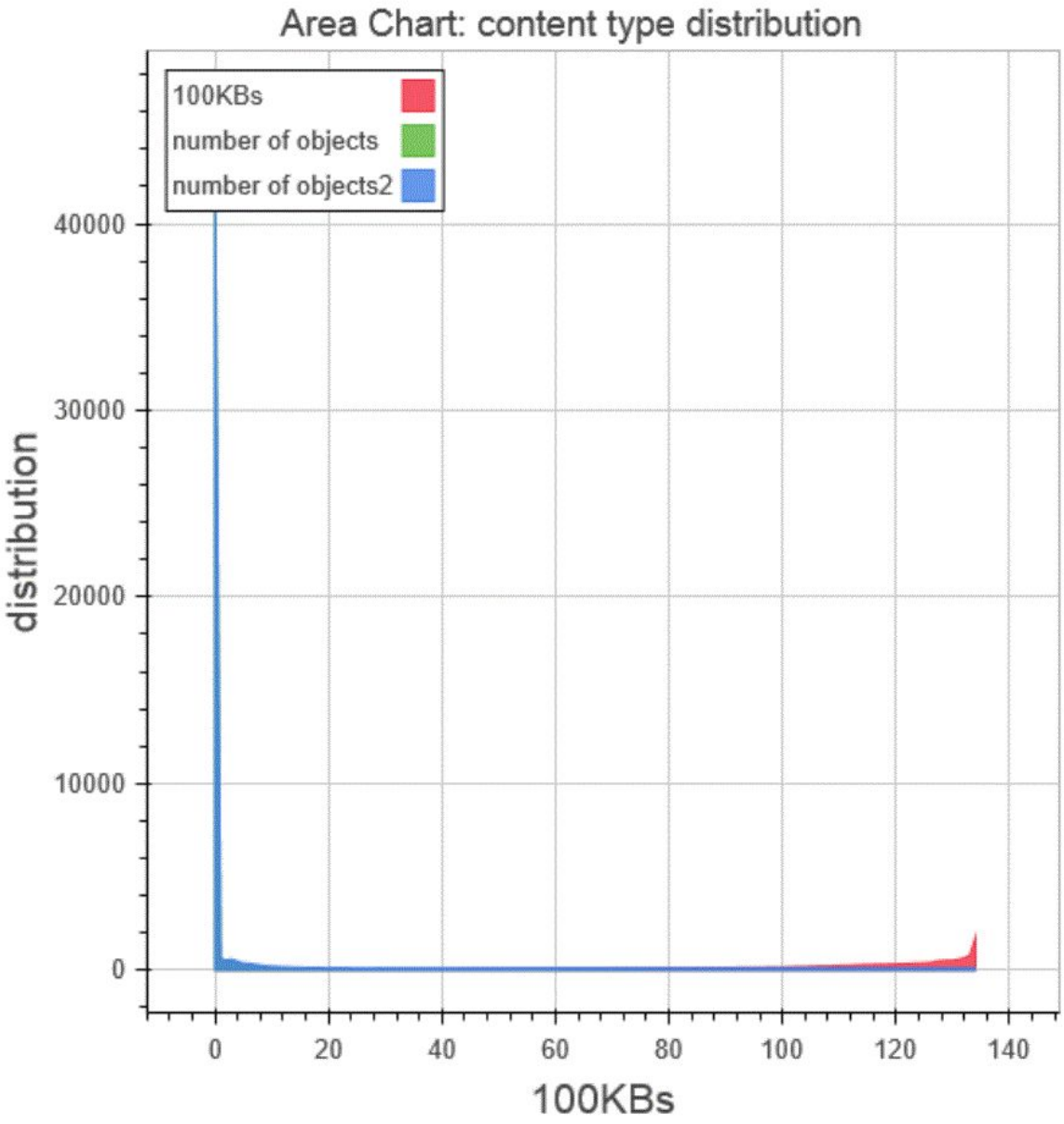

Figure 4.8: The Area Graph of the Content length Distribution

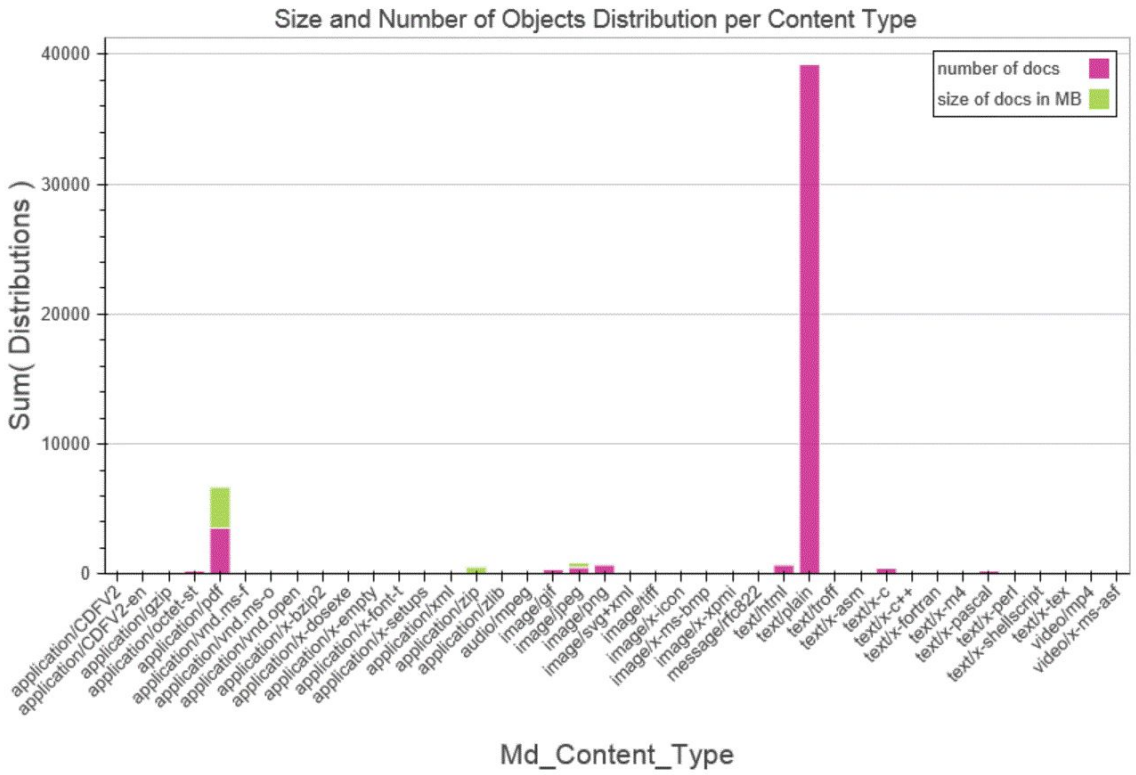

Figure 4.9: The Stacked Bar Graph of the Content Type Size and Number Distribution; It uses the same dataset as figure **??** which is the result of query in the figure **??**.

4.7.6 Donut Chart

Donut charts are like pie charts with a hole in the center. It is possible to define hole radius to any size is needed, both in percent or pixels. Similar to a pie chart, a donut chart shows the relationship of parts to a whole, but a donut chart can contain more than one data series. Each data series that you plot in a donut chart adds a ring to the chart. The first data series is displayed in the center of the chart. In fact, the primary use case for the donut chart is to show relative amount each category, within a categorical array or multiple categorical arrays, makes up of the whole for some array of values[17]. Although displaying values or percentages in data labels is very useful in a donut chart, but if you want to compare the data points side by side, you should use a stacked bar

[17] http://bokeh.pydata.org/en/0.11.0/docs/reference/charts.html#donut

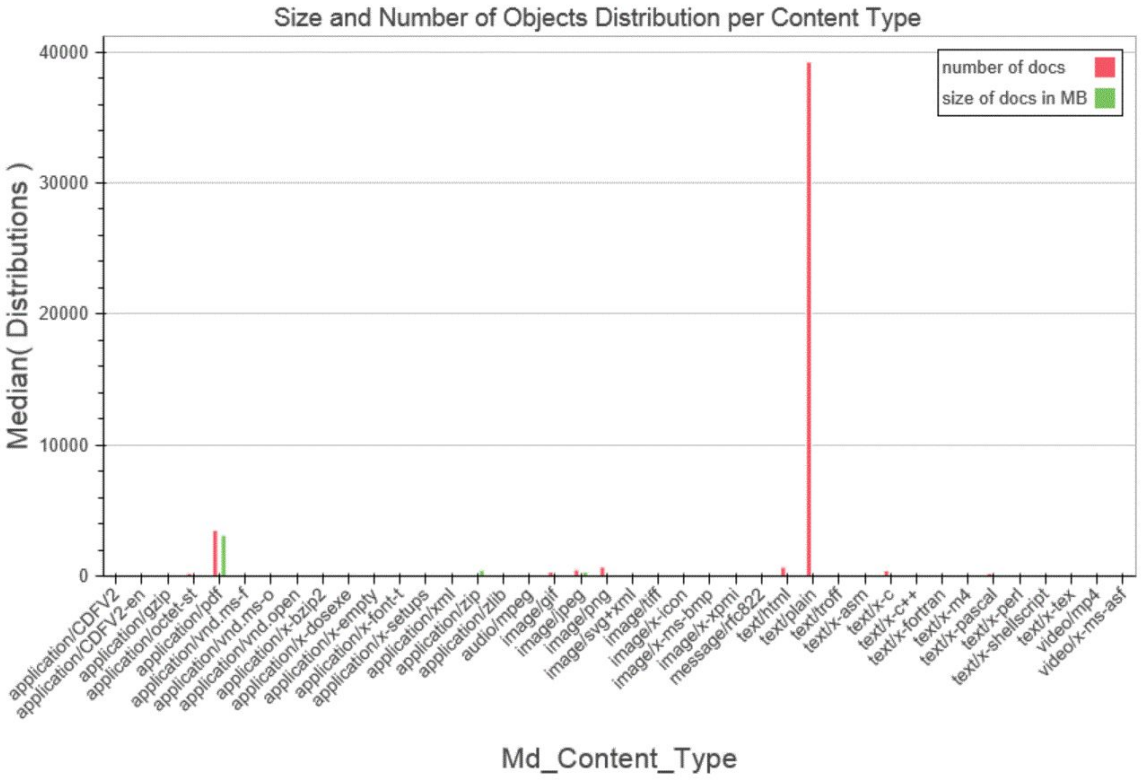

Figure 4.10: The Group Bar Graph of the Content Type Size and Number Distribution; It uses the same dataset as figure **??** which is the result of query in the figure **??**.

chart instead[18]. In the figures 4.3 and 4.12, a part of a sample data set and the resulted donut chart is shown.
In the OSECM system, if **a part to whole relationship** wants to be shown, in the structure of the data set, for each part there has to be a numerical value. There can be a descriptive column as well, but it is not necessary, since it is possible to define a suitable label for each of these columns as an attribute in the Bokeh functions. In Bokeh, it is possible to define that which columns should be shown in the circles. Moreover, it is possible to define a variable out of some columns to be plotted in one circle. This can be done in Bokeh visualization using **melt()**[19] function from **Pandas**. In the figure 4.3, an external sample data set which can be plotted in donut chart is shown.

[18] https://support.office.com/en-us/article/Present-your-data-in-a-doughnut-chart-0ac0efde-34e2-4dc6-9b7f-ac93d1783353
[19] http://pandas.pydata.org/pandas-docs/stable/generated/pandas.melt.html

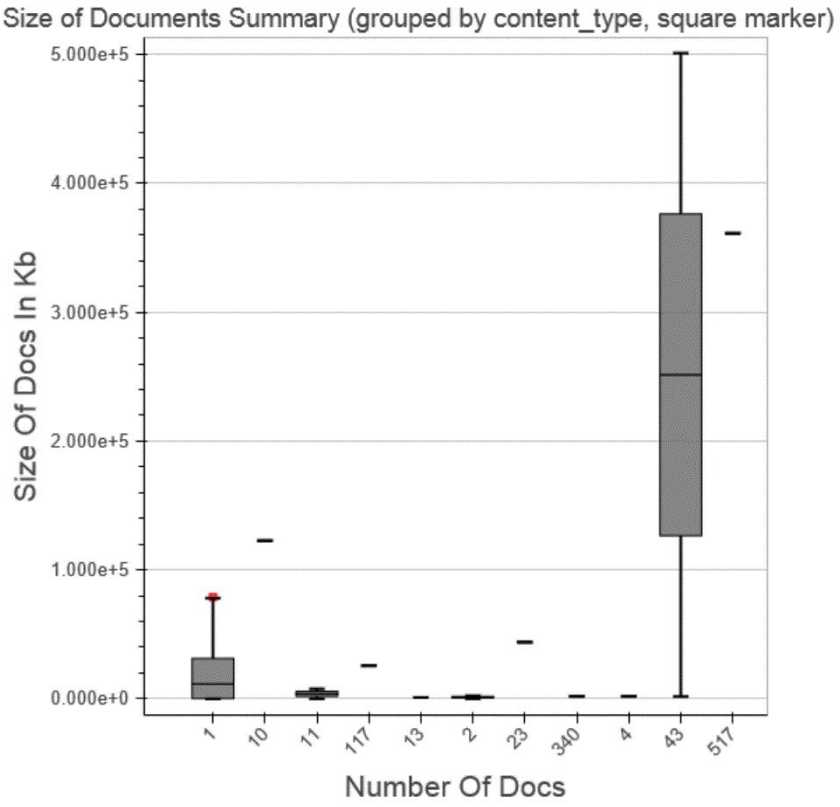

Figure 4.11: The Box Plot of the Size and Number Distribution grouped by the Object Content Type; It is the result of query in the figure **??**.

4.7.7 Use cases

Here I show some extra use cases using the sample data sets which are imported into the OSECM system. Since for analysis purpose, the data has been imported to the metadata warehouse, to construct the tables, data importer creates a row for each object and a new column for each attribute. Based on this table structure, for each of the examples, I will describe the data format and the query which extracts it. Then the visualization with a proper graph type, exposes the result data set in order to interpret the result more suitably.

Example 1: Currently, as one of the OSECM sample datasets, there are some objects imported from **Enron Email Dataset**[20] which is publicly published as a source for

[20] https://www.cs.cmu.edu/~./enron/

Table 4.3: The data set for the distribution of medals in various countries

name	abbr	medals/silver	medals/bronze	medals/gold	medals/total
Albania	ALB	0	0	0	0
Andorra	AND	0	0	0	0
Argentina	ARG	0	0	0	0
Armenia	ARM	0	0	0	0
Australia	AUS	2	1	0	3
Austria	AUT	6	1	2	9
Azerbaijan	AZE	0	0	0	0
Belarus	BLR	0	1	5	6
Belgium	BEL	0	0	0	0
Bermuda	BMU	0	0	0	0
...	...	...	...	...	...

researchers. generally each email has some attributes like "to", "from", "cc", "date" and others. In the OSECM system, a set of these metadata which are useful, has been imported to the metadata warehouse. Here, using the imported email metadata into the database, I want to show who has written the longest email. For this purpose, the query to fetch the result for this example is shown in the figure **??**.

Listing 4.5 The query to fetch the email addresses of people who has sent the largest amount of email. The top 20 senders is queried to have a better visualization in the bar graph.

```
select email.MD_from as 'Sender', sum(MD_content_length) as 'email total length' from
    MetaDataTable_email as email, MetaDataTable_SwiftInternal as swift where
    email.MD_from is not null and email.MD_containerName = swift.MD_containerName and
    email.MD_name = swift.MD_name group by email.MD_from order by sum(MD_content_length)
    desc limit 20
```

Example 2: Here, the communication among people in the email dataset is the main target. To expose who exactly emailed whom, the combination of email addresses in both **from** and **to** metadata fields for each email object should be extracted. To visualize these relations between people, using a **Graph** with a set of email addresses as the **vertexes(V)** set and the combination of sender and receiver email addresses as the **edge list(E)** is appropriate. Therefore for this example the extracted data set should have two columns each for **MD_from** and **MD_to** fields of each email. The query to extract this data set is presented in the figure **??** .

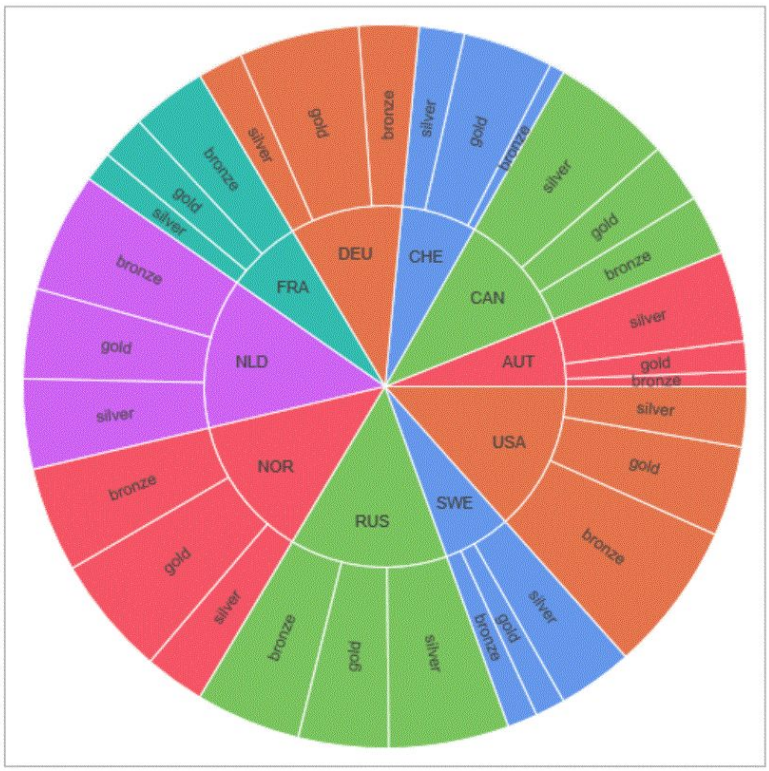

Figure 4.12: The Donut Chart of the dataset in figure 4.3

Listing 4.6 The query to fetch the list of people who are communicating with each other via email.

```
select MD_from as 'sender' , MD_to as 'receiver' from MetaDataTable_email where MD_from
    is not null and MD_to is not null
```

Example 3: Now, in the OSECM system, there are some image sample data sets with different image types which currently the OSECM system supports bmp, jpg, gif, png and tiff formats to extract metadata from them. In this use case, I want to show the relation between an image size and its resolution. Since for all of the supported image data formats, there is a metadata field which is called **MD_image_size** to keep x-resolution and y-resolution of the images, multiplying these two numbers produces a scalable number(the number of pixels) for each image object. Having the actual file size for each of the image objects, gives us another numerical property for each image. Then the system can use these two numerical properties these to plot them against each other. There is a possibility to even group these data with the different image types. Therefore

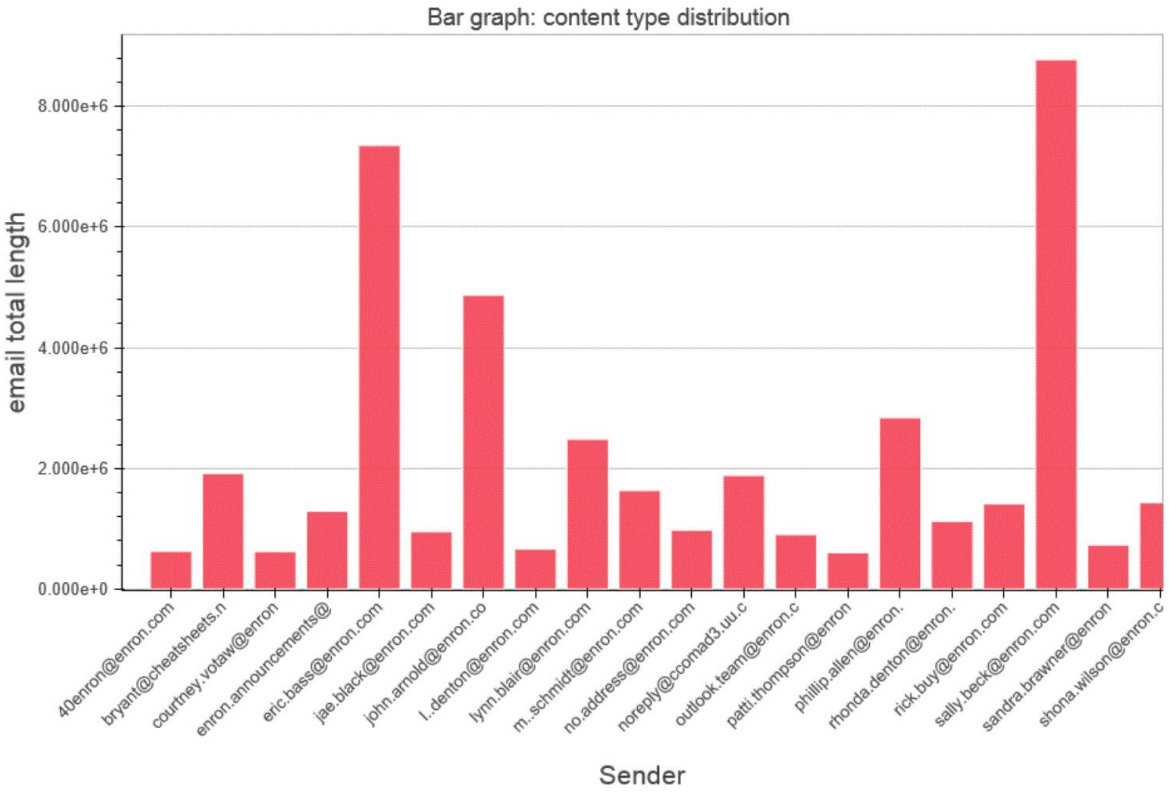

Figure 4.13: The Bar Chart to show the result set for the query in figure **??**

you can see how this distribution is for each group. For example for some of them it can be small and for the others it could be bigger. Since, currently

Example 4: Assume that there are some audio and video sample data sets. In this case, each of these objects can have some numerical properties like the length of video, file size, resolution and others. If storing these data for backing up the YouTube videos would be the main use case, then there are some other metadata fields like how many views, likes, dislikes that it has. Therefore, in this case, it is absolutely possible to plot query these metadata and plot the numerical attributes against each other using line graph, area graph or any other suitable graph.

4.8 SQLite Database

In this work, I use SQLite database as the metadata warehouse because it is relational database which supports SQL queries and it is powerful enough to support the sample analysis scenarios for this work. In fact, using a database immediately brings in the benefits of databases that have been developed over many years such as transaction support, scaling and administration capabilities. For this prototype, since I use SQLite and every thing is stored inside one database file at the end, there are not complex administration both options and difficulties. Moreover, the Node-RED server also can be conveniently connected to it. For this purpose, we have put the SQLite database file consists of the whole database system, on the Node-RED server and we added a new node type that supports the connection to the SQLite database to the list of nodes in storage category of Node-RED server.

5 Summary and Future Work

The usage of object storage instead of file systems in cloud applications is a vital approach for continuous improvement in regards to efficiency. Since object storages are supporting the metadata extensively, they are more than a plain data storage. Specially in Enterprise Content management (ECM) systems due to the fact that they rely on metadata to fulfill their main functionalities, object stores are good choices for them as a data management system on the cloud. In this thesis, I enhanced the Swift Object Store with the possibility to apply analysis scenarios specifically on top of the metadata. Often customers that want to move their applications to the cloud, require a proper level of security for their data. For this purpose, most cloud providers apply security mechanisms on both data and metadata in order to keep the system secure. The OSECM system, as an ECM system, requires an access to the metadata for content classification and organization, the security mechanisms is applied for the data. Therefore, I focus to offer the analysis features for the metadata rather than data itself.
Here, first in the Section 1.2, I investigated over different data science phases and various technologies that have been used in each phase of the data science area. As there are many tools and technologies in the industrial area for data gathering, cleaning, transforming, analysis and visualization, I discussed about some well known technologies and frameworks like Apache Hadoop, Statistical Analysis System (SAS) and others. Then in the chapter 2, first I showed the architecture of the OSECM system and describe different functionalities that it has. In the rest of the chapter, I described my contribution towards the OSECM system to provide the analysis functionalities over the available metadata. The extensive support of metadata by object stores is also highlighted through describing the possibility to add custom metadata and document classes for categorization besides the default internal metadata. Since in the OSECM system, we use a metadata warehouse as the basis for analysis functionalities, the process of fetching additional metadata and replicate them into the metadata warehouse is also discussed in this chapter. In chapter 3, the plain analysis architecture model is discussed which was the basis for the design, development and integration of the new modules with the existing OSECM system to create analysis layer. In this chapter, I discussed about different alternatives that exist to use as an analysis application component in the architecture of analysis layer. Then, I discussed that, we used Node-RED server as the analysis application which provides a high-level of flexibility to provide the possibility to define analysis queries with connecting to our metadata warehouse and any other

external sources. In this way, queries are executed inside the metadata warehouse and the result is returned back to the Node-RED. In addition, it is possible to further analyze the returned result back from the database with defining the JavaScript functions to further manipulate the result inside Node-RED server. Then, in the Section 3.1, I discussed about Bluebox UI system and its components to visualize the analysis result. Afterwards, I talked about different data structures of the result data set that can be plotted into available graph types in OSECM system. In the chapter 4, I discussed about the implementation of different module that I used to provide analysis layer for the OSECM system. These modules are Content Identifier, Metadata Extractor, Metadata Replicator, Node-RED server and Visualization Diagrams. For the metadata warehouse, I used relational database which enables the system to benefit from the advantages of SQL queries to write the analysis scenarios.

Based on this master's thesis, further enhancement of the object store can be considered through expanding analysis capabilities and providing advanced searching functionalities. For this purpose, applying more precise and complicated analysis over the metadata can be done. For instance, textual analysis, over textual datasets like emails, PDF documents and others. Furthermore, for the visualization aspects, more data structure can be defined for plotting of more complex diagrams. According to Waizenegger et al., it is also possible to investigate how the security can be provided for the metadata as well as data and provide the analysis capabilities for the non-relational data model of metadata [**OSECM**].

Bibliography

[Cha13] S. Chandrasekaran. “Becoming a Data Scientist - Curriculum via Metromap.” In: (2013). URL: http://nirvacana.com/thoughts/becoming-a-data-scientist/ (cit. on p. 12).

[Cha14] S. Chandrasekaran. “Data Science.” In: (2014). URL: www.exploringdatascience.com/the-data-science-clock/ (cit. on p. 13).

[Cor15a] E. Corporation. *Data Lake for Data Science*. EMC White Paper H14214. EMC Corporation, May 2015. URL: http://www.emc.com/collateral/white-papers/h14214-wp-emc-isilon-data-lakes-for-data-science.pdf (cit. on pp. 11, 14).

[Cor15b] I. Corporation. *IBM Watson Explorer*. IBM White Paper H14214. Software Group, Oct. 2015. URL: http://public.dhe.ibm.com/common/ssi/ecm/im/en/imb14165usen/IMB14165USEN.PDF (cit. on p. 18).

[DB13] S. Dhouib and R. Ben Halima. “Surveying Collaborative and Content Management Platforms for Enterprise.” In: (2013), pp. 299–304 (cit. on p. 25).

[DE+12] A. Divyakant, B. Elisa, et al. “Challenges and opportunities with Big Data.” In: *2012, a community white paper developed by leading researchers across the United States.[Online]. Available: http://cra. org/ccc/docs/init/bigdatawhitepaper. pdf* (2012) (cit. on pp. 13, 14, 16, 17).

[Fri08] V. Friedman. “Data visualization and infographics.” In: *Graphics, Monday Inspiration* 14 (2008), p. 2008 (cit. on p. 41).

[Hor14] Hortonworks. “A Modern Data Architecture with Apache™ Hadoop.” In: *A Hortonworks White Paper* (March 2014). URL: http://info.hortonworks.com/rs/h2source/images/Hadoop-Data-Lake-white-paper.pdf (cit. on pp. 20–22).

[Mar16] R. Marty. “Big Data Lake – Leveraging Big Data Technologies To Build a Common Data Repository For Security.” In: *Published by O’Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472* (2015.02.16). URL: http://raffy.ch/blog/2015/02/16/big-data-lake-leveraging-big-data-technologies-to-build-a-common-data-repository-for-security/ (cit. on p. 19).

[Oje14] T. Ojeda. *Practical Data Science Cookbook*. EBL-Schweitzer. Packt Publishing, 2014. URL: https://books.google.de/books?id=Mh2uoQEACAAJ (cit. on pp. 12, 13).

[R C16] R Core Team. "R: A Language and Environment for Statistical Computing." In: (2016). URL: https://www.R-project.org (cit. on p. 17).

[Vik14] S. M. .-.-. I. D. S. Vikram Andem. "Big data analysis concepts and references." In: *ISRM and IT GRC Conference Big Data Analysis* (2014). URL: http://www.slideshare.net/natemiller67/big-data-analysis-concepts-and-references?qid=c96439d8-a400-4aae-bc0a-b848ac6f8b35&v=qf1&b=&from_search=6 (cit. on p. 20).

[Wik] Wikipedia. "Data Lake." In: *Wikipedia* (). URL: https://en.wikipedia.org/wiki/Data_lake (cit. on p. 19).

All links were last followed on March 17, 2008.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature