Project Acronym	***BlueBRIDGE***
Project Title	***Building Research environments for fostering Innovation, Decision making, Governance and Education to support Blue growth***
Project Number	***675680***
Deliverable Title	***BlueBRIDGE VRE Commons Facilities: Revised Version***
Deliverable No.	***D9.2***
Delivery Date	***January 2018***
Authors	***Massimiliano Assante, Leonardo Candela, Gianpaolo Coro, Panagiota Koltsida, Nikolas Laskaris, Yannis Marketakis, Fabio Sinibaldi, Pasquale Pagano***

BlueBRIDGE receives funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 675680

DOCUMENT INFORMATION

PROJECT	
Project Acronym	BlueBRIDGE
Project Title	Building Research environments for fostering Innovation, Decision making, Governance and Education to support Blue growth
Project Start	1st September 2015
Project Duration	30 months
Funding	H2020-EINFRA-2014-2015/H2020-EINFRA-2015-1
Grant Agreement No.	675680
	DOCUMENT
Deliverable No.	D9.2
Deliverable Title	BlueBRIDGE VRE Commons Facilities: Revised version
Contractual Delivery Date	January 2018
Actual Delivery Date	March 2018
Author(s)	Massimiliano Assante (CNR), Leonardo Candela (CNR), Gianpaolo Coro (CNR), Panagiota Koltsida (NKUA), Nikolas Laskaris (NKUA), Yannis Marketakis (FORTH), Fabio Sinibaldi (CNR), Pasquale Pagano (CNR)
Editor(s)	Massimiliano Assante (CNR)
Reviewer(s)	Paolo Fabriani (ENG)
Contributor(s)	n/a
Work Package No.	WP9
Work Package Title	VRE Commons Development
Work Package Leader	CNR
Work Package Participants	CNR, UOA, FORTH
Distribution	Public
Nature	Other
Version / Revision	V1.0
Draft / Final	Final
Total No. Pages (including cover)	35
Keywords	gCube system, enabling services, VREs, data infrastructures.

DISCLAIMER

GLOSSARY

ABBREVIATION	DEFINITION
BLOB	Binary Large Objects
BlueBRIDGE	Building Research environments for fostering Innovation, Decision making, Governance and Education to support Blue growth
CKAN	Comprehensive Kerbal Archive Network
CRUD	Create Read Update Delete
CSW	Catalog Service for the Web
DMS	Data Management System
GIS	Geographic Information System
GUI	Graphical User Interface
SAI	Statistical Algorithms Importer
SDI	Spatial Data Infrastructure
SDMX	Statistical Data and Metadata eXchange
SNL	gCube Social Networking Library
VRE	Virtual Research Environment
WPS	Web Processing Service

TABLE OF CONTENT

TABLE OF FIGURES

DELIVERABLE SUMMARY

Deliverable D9.2 – "BlueBRIDGE VRE Commons Facilities: Revised Version" is the revised version of the D9.1 deliverable, intended to report the release of the BlueBRIDGE facilities for Data Access, Data Discovery, Data Storage, Data Analytics and Data Publishing. It is conceived to include the latest developments and releases that took place between the 2 versions of the reports. The deliverable is of type "Other" and the following document is intended to provide its readers with an easy-to-use description of the actual components, services and systems contributing to form the BlueBRIDGE VRE Commons Facilities. This document is structured as follows: a description for each facility is available, in conjunction with links to the project's wiki, providing more detailed descriptions and additional information for each component. This revised version of the document aims at providing a complete picture of the available facilities for Accessing, Discovering, Storing Data, Analysing and Publishing Data developed through the project's lifetime.

The deliverable is divided in two main sections. The first one follows the structure of Work Package 9; specifically, it documents the various releases and versions of the facilities available for Accessing, Discovering and Storing Data (T9.1), for Analysing Data (T9.2) and for Publishing Data (T9.3).

The second section comprises facilities for managing and using VREs that span across all the three WP9 tasks. Specifically, it includes dedicated parts documenting the various releases and versions of services for defining, creating and deploying VREs (VRE Management facilities) together with a set of applications allowing to use the VRE through a thin client (VRE Enabling facilities).

The deliverable provides, for each facility, a description, the documentation for developers and system administrators, how-to guides and usage instructions for different use cases and links to open source code and binaries. Further details and the complete documentation for the facilities are available on the gCube wiki at https://wiki.gcube-system.org/gcube/About_gCube.

The intended readers of this deliverable are (a) the community in the large willing to be informed on the solutions BlueBRIDGE offers for VRE management, Data Access, Discovery, Storage, Analytics and Publishing, and (b) the gCube developer community to know how to integrate, use and build on top of the facilities described by the document.

EXECUTIVE SUMMARY

Deliverable D9.2 – "BlueBRIDGE VRE Commons Facilities: Revised Version" is the revised version of the D9.1 deliverable, intended to report the release of the BlueBRIDGE facilities for Data Access, Data Discovery, Data Storage, Data Analytics and Data Publishing, including the latest developments and releases that took place between the 2 versions of the reports. These facilities are implemented through web services, libraries, and mediators over technologies that are offered as services, operated and made accessible through VREs.

Deliverable D9.2 is of type "Other" and it is implemented through a set of wiki pages hosted by the BlueBRIDGE project wiki at:

https://support.d4science.org/projects/bluebridge/wiki/D91_BlueBRIDGE_VRE_Commons_Facilities

At the time of its release, the deliverable covers the activities in the addressed areas performed in the first 29 months of the project.

The project activities cover all effort spent for implementing the facilities, integrating them in the gCube platform, packaging the software, testing the software in the testing infrastructure, validating it in the pre-production infrastructure, and releasing the software.

The deliverable is divided in two main parts. The first one follows the structure of Work Package 9. Specifically, it includes dedicated parts documenting the software implementing the facilities for Accessing, Discovering and Storing Data (T9.1), for Analysing Data (T9.2) and for Publishing Data (T9.3). All of the above mentioned facilities are offered through Virtual Research Environments (VREs) operated by the Infrastructure to satisfy the community needs. The second part comprises facilities for operating, managing and using, VREs. Specifically, it includes dedicated parts documenting the software implementing the facilities for defining, creating and deploying VREs (VRE Management facilities) and the software implementing a set of applications allowing to use the VRE through a thin client (VRE Enabling facilities).

Part 1 "Facilities for Data Access, Data Discovery, Data Storage, Data Analytics and Data Publishing" documents the following services:

- **Storage Manager Service:** a service providing functions for standards-based and structured access and storage of files of arbitrary size.
- **Home Library Service**: a service providing functions for managing and persisting end-users' files in the infrastructure, supporting file and folders sharing.
- **Social Networking Data Access, Discovery, Storage Facility:** a set of services providing functions for accessing, storing, indexing, and retrieving Social Networking Data (User Posts, Comments, Likes and Notifications) available in the Infrastructure.
- **Discovery Service:** a distributed service for indexing and searching which, through the use of indexing plugins, can be populated with data from various sources. It can serve as an aggregator of data, which indexes datasets under a customizable schema, offering a search facility over those datasets.
- **Data Miner Service (former Statistical Manager Service):** a service aiming to provide users and other infrastructure services with algorithms/methods to perform Data Mining operations. It implements a Cloud computing Map-Reduce approach and is able to process Big Data and to save outputs onto a collaborative experimentation space.
- **Statistical Manager Algorithms Importer Service:** a service allowing scientists to easily and quickly import R language scripts onto the above described DataMiner which, in turn, publishes these

scripts as-a-Service and manages multi-tenancy and concurrency. Additionally, it allows scientists to update their scripts without following long software re-deploying procedures each time.

- **The Spatial Data Infrastructure Service (former GIS Interface Publisher service):** a service acting as a main control point for the entire Spatial Data Infrastructure (SDI). It manages configuration of available GIS Services and their registration into the infrastructure's Information System.
- **Resource Catalogue service:** a service that makes infrastructure's products accessible by providing tools to streamline publishing, sharing, finding and using their data (and metadata). All products are accompanied with rich descriptions capturing general attributes, e.g. title and creator(s), as well as usage policies and licences.

Part 2 "Facilities for VRE Management and Usage" includes the following:

- **VRE Management facilities:** a set of services and applications providing functions for defining, creating and deploying VREs. These services support VRE Designers and Managers through graphical user interfaces to instruct the infrastructure about the expected features of the desired VRE as well as allowing to easily update the VRE once defined and operational.
- **VRE Enabling Portlets:** a set of interaction-oriented services and front-end applications providing functions to manage VRE Users and to support them to collaborate, cooperate, and exchange content/information by using social-media-like tools.

The Social Networking Data Discovery Service, the Resource Catalogue, the Data Miner Service, the Statistical Manager Algorithms Importer Service (SAI), the Discovery Service and the Spatial Data Infrastructure Service are completely new services entirely designed and developed in the context of BlueBRIDGE. The rest of the Facilities and Portlets listed above are components that pre-exist BlueBRIDGE. In the context of BlueBRIDGE those components have been re-designed to be adapted to the evolving technologies and to deliver: (i) personalized, interactive and collaborative content, (ii) distributed, loosely coupled, multi-platform architectures supporting huge volumes of data as well as (iii) scaling out rather than scaling up. Moreover, all the components have been revised to conform to the new gCube Authorization Framework, described in the deliverable D10.1 BlueBRIDGE Resources Federation Facilities.

The deliverable provides, for each facility, a description, the documentation for developers and system administrators, how-to guides and usage instructions for different use cases and links to open source code and binaries. Further details and the complete documentation for the facilities are hosted on the gCube wiki at https://wiki.gcube-system.org/gcube/About_gCube.

The intended readers of this deliverable are (a) the community in the large willing to be informed on the solutions BlueBRIDGE offers for Accessing, Discovering, Storing, Analysing Data and Publishing Data as well as for Virtual Research Environment management, and (b) the gCube developers community to know how to integrate, use and build on top of the facilities described by the document.

1 Facilities for Data Access, Data Discovery, Data Storage, Data Analytics and Data Publishing

Facilities for Data Access, Data Discovery, Data Storage, Data Analytics and Data Publishing are provided by the following components:

- Storage Manager Service (cf. Sec. 1.1) for files storage;
- Home Library (cf. Sec. 1.2) for objects management;
- Social Networking Services (cf. Sec. 1.3) for communication;
- Discovery Service (cf. Sec. 1.4) for discovery;
- Data Miner Service (cf. Sec. 1.5) for analytics;
- Statistical Manager Algorithms Importer Service (cf. Sec. 1.6) for adding algorithms to the analytics platform;
- Spatial Data Infrastructure Services (cf. Sec. 1.7) for storing, accessing, discovering, and publishing geospatial datasets;
- Resource Catalogue Service (cf. Sec. 1.8) for storing and publishing information (metadata) on resources of defined types (including datasets) and supporting the discovery published resources and the access to them.

1.1 Storage Manager Service

A service providing functions for standards-based and structured access and storage of files of arbitrary size is a fundamental requirement for a wide range of system processes, including indexing, transfer, transformation, and presentation. Equally, it is a main driver for clients that interface the resources managed by the system or accessible through facilities available within the system.
The Storage Manager Service abstracts over the physical storage and is capable of mounting several different store implementations (by default clients can make use of the MongoDB store) presenting a unified interface to the clients and allowing them to download, upload, remove, add and list files or unstructured bytestreams (binary objects). These have owners and owners may define access rights to files, allowing private, public, or shared (group-based) access.
All the operations of this service are provided through a standards-based, POSIX-like API which supports the organisation and operations normally associated with local file systems whilst offering scalable and fault-tolerant remote storage

As shown in Figure 1. the core of the Storage Manager service is a software component named Storage Manager Core exposing the above-mentioned storage manipulation API, abstracting the specificities of the actual physical storage. The Storage Manager Wrapper instead is a software component used to discover back-end information from the IS Collector service of the D4Science Infrastructure Information System. The separation between these 2 components is necessary to allow the usage of the service in different contexts other than the D4Science Infrastructure. Overall, the service allows Content/Ready/Update/Delete (CRUD) operations on binary objects as well as Move and Copy operations.

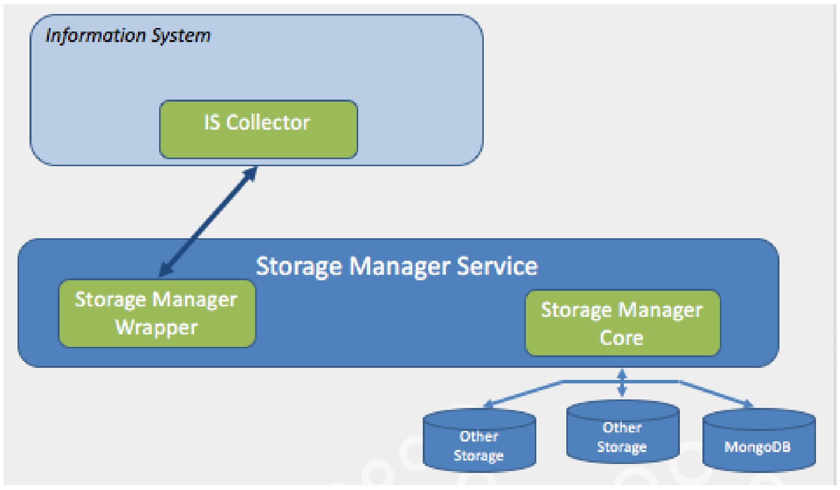


Figure 1. Storage Manager Service Architecture

Further details and information can be found in the gCube Wiki page related to the service at the web address:
https://gcube.wiki.gcube-system.org/gcube/Storage_Manager

This service pre-exists BlueBRIDGE. During the first period it has been re-designed to adapt to the evolving technologies.

1.2 Home Library Service

The Home Library service exposes functions for managing and persisting end-users' files in the infrastructure, supporting file and folders sharing. In particular, the service provides a remote (Cloud) folder-based file system, supporting sharing of folders and different item types (ranging from binary files to information objects representing, for instance, tabular data, workflows, distribution maps, statistical algorithms).
The Home Library service delivers the concept of personal Workspace, where users can collaborate, share information, and access resources using folders. Users indeed are one fundamental entity managed by this service, they are assigned a Role and a Role is what defines their privileges. In order to do so the Home Library service provides support for Access Control Management over the repository content, enabling the following:

- Users & groups management;
- Privilege discovery, i.e. determining the privileges that a user or group has in relation to a given content;
- Access control policies assignment, i.e. setting the privileges that a user or group has in relation to given content using access control policies.

As shown in Figure 2. this service relies on 2 different storage technologies: (i) an Apache Jackrabbit content repository, properly replicated, is used to store the metadata of the items being stored (by means of a RMI Protocol) while the actual payload of these items is stored in a (ii) MongoDB Cluster (by means of the Storage Manager service facilities introduced in 1.1).
The Home Library service exposes the contents as HTTP resources, fostering a RESTful style of application architecture. Any (authorised) Service can contact the Home Library service via a Standard HTTP Client, or via a Java Home Library Client provided with this service.

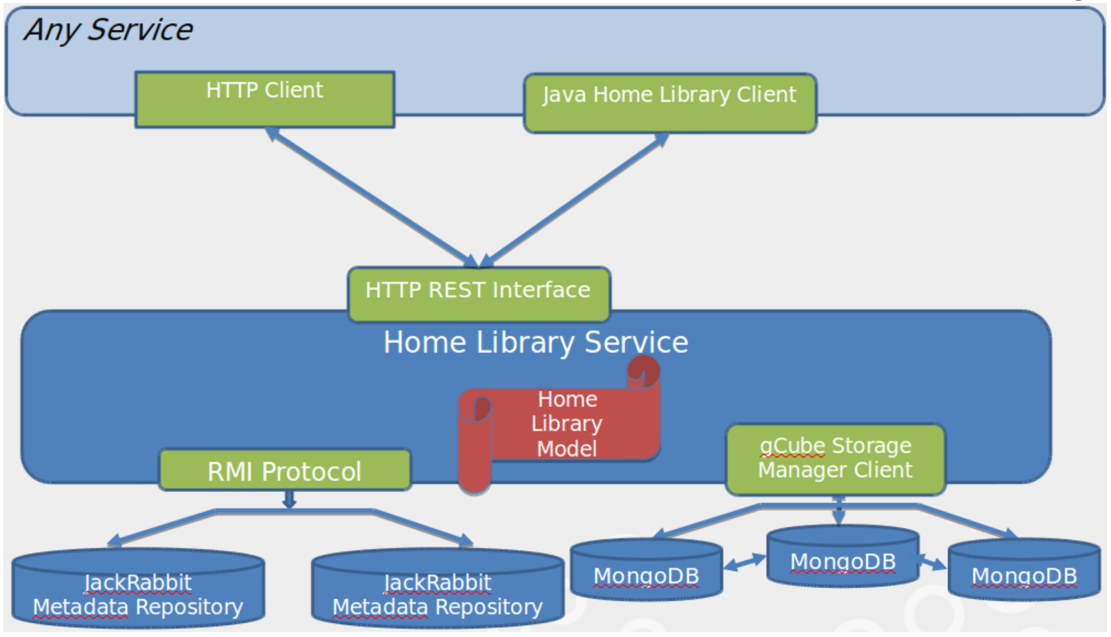


Figure 2. Home Library Service Architecture

Further details and information can be found in the gCube Wiki page related to the service at the web address:
https://wiki.gcube-system.org/gcube/Home_Library_2.0_API_Framework_Specification

This service pre-exists BlueBRIDGE. During the period it has been re-designed to adapt to the evolving technologies.

1.3 Social Networking Data Access, Discovery, Storage Facility

A set of services providing functions for accessing, storing, indexing and retrieving Social Networking Data (User Posts, Comments, Likes and Notifications) available in the Infrastructure.
The social networking services exploit a NoSQL data store as storage technology (Apache Cassandra cluster) and an Elasticsearch cluster for supporting the discovery facility.

1.3.1 Social Networking Data Access and storage service

All the services rely on a core Java library called gCube Social Networking Library (SNL). SNL offers access methods, such as post retrieval/creation/deletion, comment retrieval/creation/deletion, notifications etc. The SNL pre-exists BlueBRIDGE, during the period it has been re-designed to adapt to the evolving technologies. Detailed information about SNL can be found in the related WIKI page at the following web address: https://wiki.gcube-system.org/gcube/Social_Networking_Library

1.3.2 Social Networking Data Discovery service

The Social Networking Data Discovery service is a completely new service developed during the reporting period. It has been specifically designed to offer full-text search capabilities over the social networking data in the context of BlueBRIDGE.

The engine enabling full-text search is ElasticSearch, a highly scalable, distributed, open source search and analytics engine based on the Apache Lucene software library. It runs over one or more cluster nodes and is reachable over http(s) protocol. ElasticSearch allows to organise documents in one or more indices/types according to their schema. This schema can be defined in JSON format. Elasticsearch features a near real-

time search platform meaning that there is a low latency (normally one second) from the time a document is indexed until the time it becomes searchable.
The glue between Cassandra and ElasticSearch is a component based on the gCube SmartExecutor service. The SmartExecutor service allows to execute custom "Tasks" in the form of plugin components, and monitor their execution status. Each instance of the SmartExecutor service can run the "Tasks" related to the plugins available on such an instance. The SmartExecutor plugin related to the Social Networking Data Discovery is named social-data-indexer plugin.

The main goal of the Social Networking Data discovery service is to let the users quickly search over this (potentially huge) amount of data, taking into account the data they are allowed to access: a user is allowed to see only the data of the VREs in which she is present. In order to do that, a client Java library has been developed. It receives the query submitted by the user and returns the list of posts belonging to the user's VREs, if any, sorted according to the score they reached.

1.3.2.1 Key features

In order to understand which are the key features of the Social Networking Data Discovery facilities, it is necessary to understand what are the data that Apache Cassandra stores and how we can help users to quickly retrieve information.

These data types are:

- Users' posts;
- Users' comments;
- User's file attachments metadata (the payload of such attachments is stored into a different database).

The full-text search focuses on the data types described above. A single user's post can be composed by the following elements:

- Post's text: the content of the post (mandatory);
- Post's author: who made the post (mandatory);
- zero or more comments to the post, or zero or more comments' texts and zero or more comments' authors;
- zero or more attachments (e.g. pdf, images, csv files);
- a VRE within the infrastructure in which the post/comments were published (mandatory).

A post with the related text, comments, authors and attachments will be called an enhanced post.
Any user can:

- retrieve posts by author (of both post/comments);
- retrieve posts by content (of both post/comments);
- retrieve posts by attachments' names.

In fact, these are the current posts fields that are discoverable. In the following "Use Cases" paragraph, we are going to discuss each scenario.

1.3.2.2 Use cases

As stated above, the discovery service allows to retrieve a post when the query matches at least one among its content, author or comment, comments' author or attachment name field.

The used approach is to make a MultiMatch Query, *i.e*. the final score of a post evaluated as if the query matches against each field of this document to evaluate a partial score. The partial scores are then summed up to get a final score for the document w.r.t. the query. The most_fields type of MultiMatch query makes sense when we are querying different document's fields analyzed in different ways, as it is our case.

A list of possible use cases includes the following:

1. Search all posts whose author is a specified user, e.g. "John Smith": in this case the full name "John Smith" can be inserted into a search textbox;
2. Search all posts in which there are comments by a specified user, e.g. "John Smith": in this case the full name "John Smith" is inserted into the search textbox;
3. Search all posts having a file attached with name "report.pdf": in this case she can insert "report" or "report.pdf" into the search textbox;
4. Search for all the posts with a .doc document attached: in this case ".doc" is inserted into the search textbox;
5. Search for a specific topic, e.g. "social data indexing": in this case she inserts "social", "social data", "social indexing" or the other possible combinations are used to search the posts.

1.3.2.3 Design

There are different components involved for making the Social Networking Data search feature work. In the following we present how ElasticSearch has been prepared and how it interacts with the Social-Data-Indexer plugin. Specifically, we need to tell to the ElasticSearch engine what are the fields of the document that are needed to be indexed, so that the engine is able to create the inverted indices and all the related structures needed to speed up the query and the retrieval phases. Moreover, we need to instruct it on how it has to analyse the documents at indexing time (when a document is inserted or updated), and at query time, (the query is analyzed for that specified field). This is an important aspect because it influences the accuracy of the results.

Our interest is making the following fields searchable:

- Post's description;
- Post's author;
- Comment's description;
- Comment's author;
- Feed's context and
- Attachment's name.

To this extent, we need to specify the document's schema of the objects pushed to ElasticSearch. We underline which fields must be indexed (and how must be analyzed) and the ones won't be analyzed at all, therefore won't be indexed/searchable. The "mapping" schema used as well as detailed information on the fields that are indexed and analyzed are available in the related wiki page:

https://wiki.gcube-system.org/gcube/Social_Networking_Data_Discovery - Mapping_Schema

The analyzer instead is composed by:

1. a CharFilters that comes first of the Tokenizer to delete/transform chars (e.g. the html_strip filter to delete html tags).
2. a Tokenizer used to split words (e.g., the Whitespace tokenizer);
3. zero or more TokenFilters used to modify/remove/add tokens starting from the stream of tokens received by the Tokenizer (e.g. the Standard Token filter, the AsciiFolding Token filter, etc), and

4. ElasticSearch offers many precooked Tokenizers, Filters and CharFilters and Analyzers. New analyzers can be created by composing the different sub pieces. Moreover, as stated above, different analyzers can be used for the same field at index time and at query time.

To choose the analyzer that best fits the field to which it will be applied, one needs to know the data to be indexed. For instance, the author full name is supposed to be composed by:

1. Name;
2. Middle Name;
3. Last Name.

These are separated by whitespaces and can contain characters such as "ä", "ö" etc. Since it is very difficult that a person, looking for someone else's posts, exactly remembers the name of that person it is helpful to transform that chars into the ASCII equivalents ones. So that "ä" becomes "a", "ö" becomes "o" etc. This operation must be performed both at query and index time. Thus, an analyzer for this field can be declared this way:

```
"author_analyzer": {
   "tokenizer": "whitespace",
      "filter": [
         "asciifolding",
         "lowercase"
      ]
 }
```

The strings are also lowercase so that searching for "John Smith" or "john smith" makes no difference at all (both will be converted to "john smith" and tokenized to "john" and "smith").

A more complex analyzer has been defined for the post/comment description. This description could contain in principle html tags (e.g. <p>This is a paragraph</p>), hashtags (e.g. #elasticsearch), non ASCII chars, and a lot of meaningless words (e.g. "this", "a", "an", "the", also known as 'stopwords'). To properly analyze this field we need to consider that:

- hashtags must be retained (default analyzers remove the char "#" but we want to preserve them)
- html tags are useless, thus must be removed;
- stopwords are meaningless, so it is better to remove them (they would also increase space usage and document retrieval time).

Furthermore, suppose there is an hashtag “#elasticsearch”, and we are interested in retrieving document that have this hashtag but, unfortunately, we do not remember the entire hashtag. A way to help the user is to store within the index the word "#elasticsearch", but also "#elas", "#elastic" and so on. This is what the 'edge-n-gram' tokenizer does: it needs two parameters to be defined, the minimum length of the word to be generated and the maximum one. For instance, if we use an edge-n-gram filter with values [2, 20], the word "elasticsearch" generates this stream of tokens: “el, ela, elas, and so on, up to elasticsearch”. It is reasonable to apply the edge-n-gram filter at indexing time, but not at query time. Doing it at query time would enlarge the time needed for the query to be evaluated as well as to sort the large amount of documents that would match.

We defined the “description comment post index analyzer” and the “description comment post search analyzer” as follows:

```
"description_comment_post_index_analyzer": {
  "type": "custom",
  "char_filter": [
    "html_strip"
  ],
  "tokenizer": "whitespace",
  "filter": [
    "lowercase",
    "hashtag_filter",
    "edge_ngram",
    "asciifolding",
    "stopwords_eng_remover"
  ]
}

"description_comment_post_search_analyzer": {
  "type": "custom",
  "char_filter": [
    "html_strip"
  ],
  "tokenizer": "whitespace",
  "filter": [
    "lowercase",
    "hashtag_filter",
    "asciifolding",
    "stopwords_eng_remover"
  ]
}
```

The hastag_filter simply specifies that when a word containing "#" is found, this symbol must be considered as a word delimiter, thus it won't be deleted. Finally, to perform a full-text search over attachment's name we used a pattern tokenizer that uses a regular expression to split the original filename, so that most punctuation is removed. After that, a edge-n-gram token filter is used, followed by lowercase and asciifolding filters.

The social-data-indexer-se-plugin is a SmartExecutor plugins that runs every T minutes in order to push back the Cassandra data, prepared according to the document schema, into the ElasticSearch engine. Furthermore, it removes documents related to enhanced feeds no longer present on Cassandra and updates changed feeds. We decided to perform this synchronization task every T minutes and avoid to push every single change to the Elastic cluster for different reasons:

1. to reduce the load of both clusters;
2. to allow to decouple ElasticSearch and Cassandra;
3. to not require other data structures that would have been needed for supporting document deletion;
4. to let users expect to be helped in retrieving old feeds and not the most recent ones.

The SmartExecutor takes care of the lifecycle of the plugin and each plugin's execution can be monitored, the simplified code is available in the related wiki page at: https://wiki.gcube-system.org/gcube/Social_Networking_Data_Discovery - SmartExecutor_SocialDataIndexer_Plugin.

1.3.2.4 ARCHITECTURE

Figure 3 shows how the different components interact each other and the roles they have. The Cassandra Cluster stores all raw information we need so it is queried by the Social-Data-Indexer plugin every T minutes to fetch new documents. The Social-Data-Indexer plugin builds up the enhanced posts, pushes them to the ElasticSearch cluster by using the ElasticSearch Bulk APIs and deletes the documents that refer to no longer present posts in Cassandra.
The ElasticSearch Client Library (ESCL) is deployed into the Infrastructure Portal (Gateway in the Figure). A user interested in retrieving documents, transparently uses it. When one or more matching documents to the query are retrieved by ElasticSearch, the JSON document is mapped back to the EnhancedPost class in Java , so that a list of those class beans is returned.

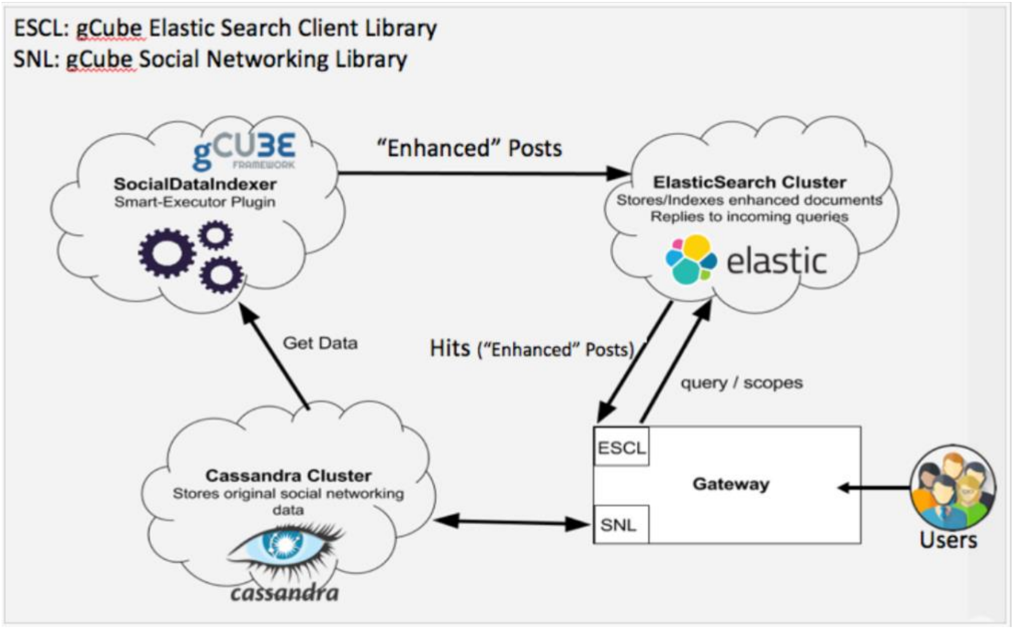


Figure 3. Social Networking Data Discovery Architecture

1.3.2.5 API AND USAGE EXAMPLES

API and Usage Examples are available in the related wiki pages at the web addresses:

- https://wiki.gcube-system.org/gcube/Social_Networking_Data_Discovery - API
- https://wiki.gcube-system.org/gcube/Social_Networking_Data_Discovery#Usage.2FExamples

1.3.3 SEARCH FRAMEWORK

The gCube Search Framework is a completely modular and distributed indexing and searching framework, which through the use of indexing plugins, can be populated with data from various sources. It can serve as an aggregator of data, which indexes datasets under a customizable schema and offering a search facility over those datasets.

It consists of several components:

- A distributed indexing mechanism - based on the ElasticSearch framework, which runs within Apache Tomcat containers and offers a customized REST API to index new datasets and acquire through highly customizable queries the stored datasets;
- An Index/Search administration portlet, allowing authorized end users to customize/maintain the indexed datasets (i.e. change field names through aliasing, change search result facets, show/hide index fields from result sets, re-index datasets with different configuration, delete indexed datasets, etc);
- A Search portlet to allow users to submit in a very simplistic way, complicated search queries to fetch the desired results in a user-friendly view format;
- An OAI-Publisher service to allow publishing the stored data under the oai-pmh protocol;
- A Java client library to allow any other Java-based app to communicate with the framework (search data, index data, change infrastructure configuration, etc).

A simplified architecture diagram of the Index and Search framework is depicted below.

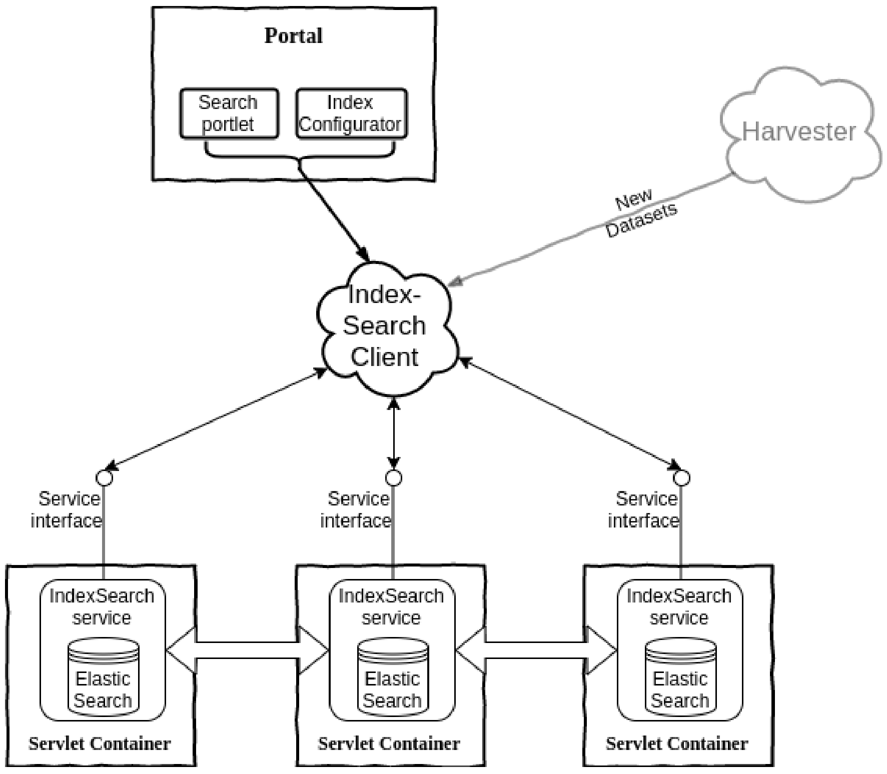


Figure 4. Simplified architecture diagram of the Index and Search framework

1.4 DATA MINER SERVICE (FORMER STATISTICAL MANAGER SERVICE)

The BlueBRIDGE data processing platform (named DataMiner [7]) is an open-source computational system based on the gCube system. This platform is fully integrated with the D4Science e-Infrastructure, which underlies the BlueBRIDGE e-infrastructure, and has been conceived to meet new Science paradigms requirements. These paradigms are born in the last decades to promote collaborative experimentation and publication of scientific findings, and are currently evolving in the context of Big Data. They foster the open publication of results, findings and documents related to scientific research, and promote Computer Science systems using collaborative approaches.

DataMiner was born in this context and supports a number of new paradigms-related requirements. This system is able to interoperate with the services of the D4Science e-Infrastructure; it uses the Web Processing Service (WPS) [8] standard to publish the hosted processes and saves the provenance of an executed experiment using the standard Prov-O [9] ontological representation. DataMiner implements a Cloud computing MapReduce approach and is able to process Big Data and to save outputs onto a collaborative experimentation space (the D4Science Workspace), which allows users to share computational information with other colleagues. DataMiner was also conceived to execute processes provided by communities of practice in several domains, reducing integration effort at the same time. The DataMiner deployment is fully automatic and is spread across different machines providers (including the European Grid Infrastructure Federated Cloud system [10]).

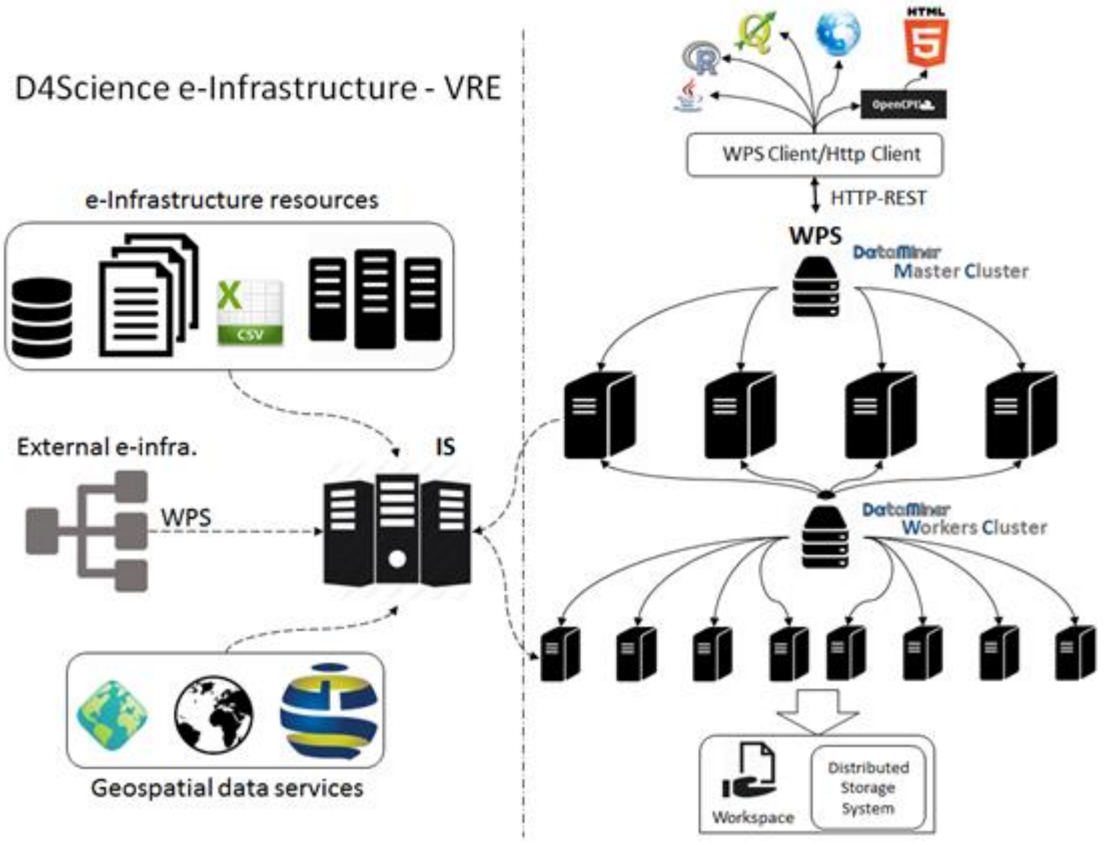


Figure 5. Architecture of the DataMiner data processing system

The DataMiner (DM) architecture is made up of two sets of machines (clusters) that operate in a Virtual Research Environment: the Master and the Worker clusters. In a typical deployment scenario, the Master cluster is made up of a number of powerful machines (e.g. Ubuntu 14.04.5 LTS x86 64 with 16 virtual CPUs, 32 GB of random access memory, 100 GB of disk) managed by a load balancer that distributes the requests uniformly to the machines. Each machine is endowed with a DM service that communicates with the D4Science Information System (IS), i.e. the central registry of the e-Infrastructure resources, to notify its presence and capabilities. The balancer is indexed on the IS and is the main access point to interact with the DMs. The machines of the Worker cluster have less local computational power (e.g. Ubuntu 14.04.5 LTS x86 64 with 2 virtual CPUs, 2 GB of random access memory, 10 GB of disk) and serve Cloud computations. DM is based on the 52North WPS service implementation[1], but extends it to meet the D4Science e-Infrastructure requirements. It is developed with Java and the Web service runs on an Apache Tomcat instance endowed with gCube system libraries. Further, it offers a development framework to integrate new algorithms and to interact with D4Science.

When a WPS request comes to the Master cluster balancer, it is distributed to one of the cluster services (Master DM). The DMs host processes provided by several developers. In particular, two kinds of algorithms are hosted: "local" and "Cloud" algorithms. Local algorithms are directly executed on the Master DMs and possibly use parallel processing on several cores and a large amount of memory. Instead, Cloud algorithms use distributed computing with a Map-Reduce approach and rely on the DMs in the Worker cluster (Cloud nodes).

With respect to the standard 52North implementation, DM adds a number of features. First, it returns a different list of processes according to the VRE in which the service is invoked. When an algorithm is installed on a DM, it is also indexed on the IS as a resource. Thus, an e-Infrastructure manager can assign it to a number of VREs. When invoked in a VRE, DM returns only the subset of hosted processes that have

[1] http://52north.org/communities/geoprocessing/wps/

been assigned to that VRE. On the other hand, one may also want to create multidisciplinary VREs with algorithms belonging to different domains.

Adopting the WPS standard in a Cloud computing system allows a number of thin clients to use the processes. Third-party software (e.g. the well-known QGIS and ArcMap software for geospatial data manipulation) can be able to retrieve the capabilities of a WPS service and to run remote processes. Further, through the gCube framework, DataMiner offers clients for R and Java[2] and the WPS service can manage HTTP-GET requests. Thus, a process can be also invoked using a Web browser, which makes it easy to repeat an experiment. Finally, an OpenCPU[3] instance is provided in D4Science, which transforms WPS objects into Javascript objects and allows for fast building of HTML applications.

The DataMiner computations can take inputs from the D4Science Workspace. Inputs can also come from Workspace folders shared among several users. This fosters collaborative experimentation already at the input selection phase. Inputs can also come from external repositories, because a file can be provided either as an HTTP link or embedded in a WPS execution request. The outputs of the computations are written onto the D4Science Distributed Storage System and are immediately returned to a client at the end of the computation. Afterwards, an independent thread also writes this information on the Workspace. Indeed, after a completed computation, a Workspace folder is created which contains the input, the output, the parameters of the computation, and a provenance document summarizing this information. This folder can be shared with other people and used to execute the process again. Thus, the complete information about the execution can be shared and reused. This is the main way by which DataMiner fosters collaborative experimentation.

The DM processes can access to the resources available in a VRE by querying the IS; for example, it is possible to discover geospatial services, maps, databases and files. The DM Java development framework simplifies the interaction with the IS. Since the IS interface is HTTP REST, it can be managed by the processes directly. Further, the DM development framework provides methods to transform heterogeneous GIS formats into a numeric matrix and thus simplifies the effort to process geospatial data. DataMiner can also import processes from other WPS services: if a WPS service is indexed on the IS for a certain VRE, its processes descriptions are automatically harvested, imported, and published among the DM capabilities for that VRE. During a computation, DM acts as a bridge towards the external WPS systems. Nevertheless, DM adds provenance management, authorization, and collaborative experimentation to the remote services. The processes currently hosted by DataMiner are written with the Java, R, Fortran, C, Octave, Linux-Shell, Windows-Batch, and Python programming languages and have been provided by developers with heterogeneous expertise (e.g. biologists, mathematicians, agronomists, physicists, data analysts etc.).

[2] https://wiki.gcube-system.org/gcube/How_to_Interact_with_the_DataMiner_by_client

[3] https://www.opencpu.org

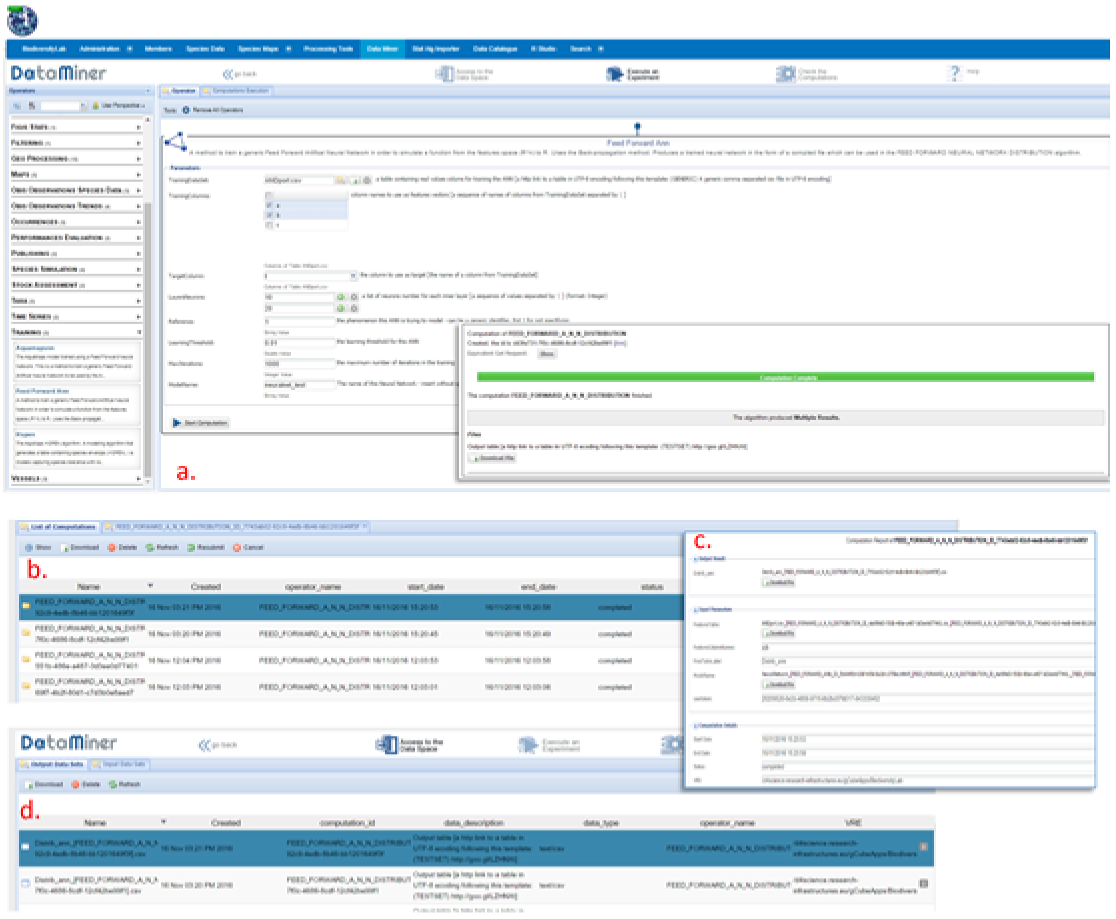


Figure 6. Graphical User Interface of the gCube DataMiner system

DataMiner offers a Web GUI to the users of a VRE (Figure above). On the left panel (Figure 6.a), the GUI presents the list of capabilities available in the VRE, which are semantically categorised (the category is indicated by the process provider). For each capability, the interface calls the WPS DescribeProcess operation to get the descriptions of the inputs and outputs. When a user selects a process, in the right panel, the GUI on-the-fly generates different fields corresponding to the inputs. Input values can be selected from the Workspace (the button associated to the input opens the Workspace selection interface). The “Start Computation” button sends the request to the DM Master cluster, which is managed as explained in the previous section. The usage and the complexity of the Cloud computations are completely hidden to the user, the type of the computation is reported as a metadata in the provenance file. In the end, a view of the Workspace folders produced by the computations is given in the “Check the Computations” area (Figure 6.b), where a summary sheet of the provenance of the experiment can be obtained (“Show” button, Figure 6.c). From the same panel, the computation can be also re-submitted. In this case, the Web interface reads the Prov-O XML information associated to a computation and rebuilds a computation request with the same parameters. The computation folders may also include computations executed and shared by other users. Finally, the “Access to the Data Space” button allows obtaining a list of the overall input and output datasets involved in the executed computations (Figure 6.d), with provenance information attached that refers to the computation that used the dataset.

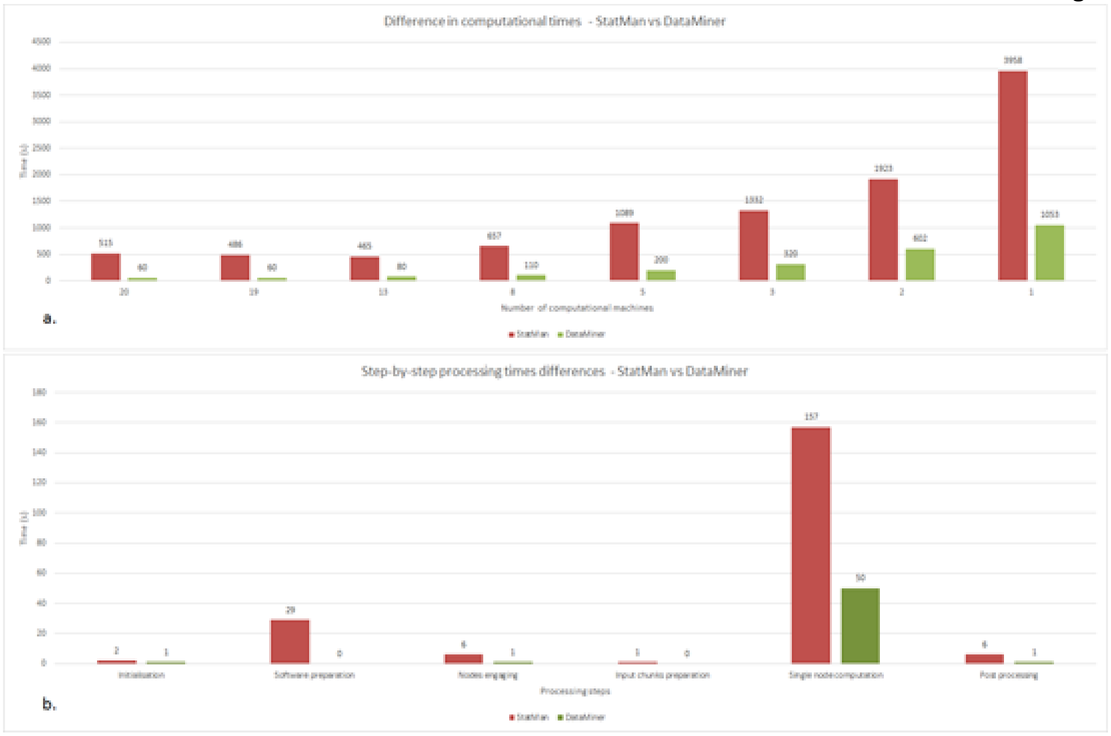


Figure 7. Performance comparison between the DataMiner and StatMan cloud computing systems

The architectural choice and the use of standards also improve the performance of DataMiner with respect to other platforms. Indeed, the performance of DataMiner can be compared with the one of another Cloud computing system (StatMan [11]) having an architecture that is quite similar to other state-of-the-art systems. In StatMan, a users' request is queued until one of the Master machines is ready to process it. Further, the Cloud computing machines are separate services that execute processes as standalone, compiled software (using the same approach as Apache Hadoop). Thus, StatMan requires a process to be prepared also in a standalone version. On top of the Cloud machines, a queue service hosts messages corresponding to the chunks of a computation, which are consumed by the Workers one at time. Since this approach is common to several distributed computing systems, StatMan may represent also other systems to a certain extent. When compared on the execution of the same process (BiOnym [12] taxonomic search) using several machines, the difference of performance can be evaluated both in the overall computational times and in the response to each step of the computation. Overall, DM gets faster and faster with respect to StatMan as more nodes are used: with 20 nodes, the time reduction is ~88%, whereas with one node it is ~73%. Indeed, the average single-node computation is faster on DM, because the process is not executed in a standalone fashion. The DM post-processing phase is faster too, because the output is written on the D4Science Storage System directly, whereas the Workspace is used only after and independently of the computation.

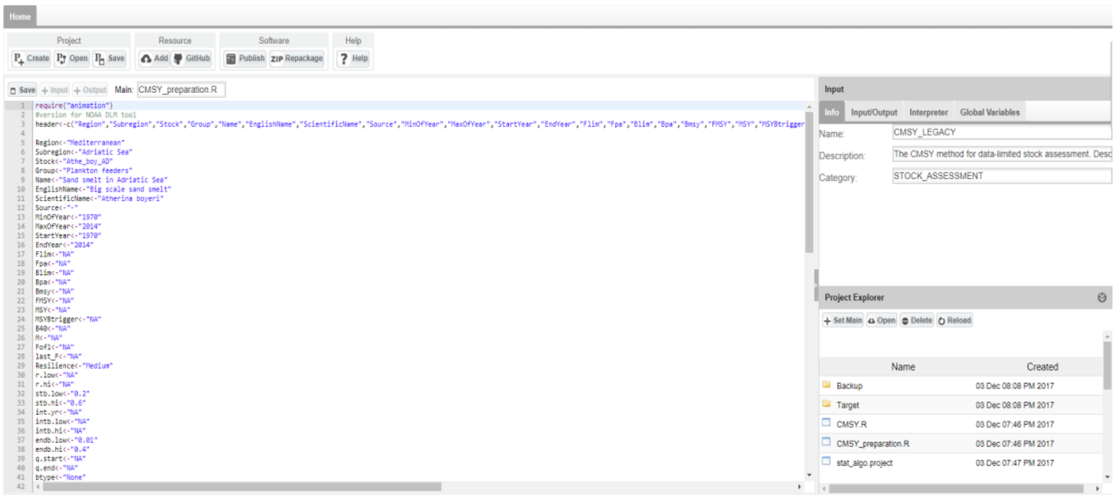
The DataMiner services use the security services of the D4Science e-Infrastructure and require a user token[4] for each operation. This token is passed via basic HTTPS-access authentication, which is supported by most WPS and HTTP(S) clients. The token identifies both a user and a Virtual Research Environment and this information is used by DM to query the IS about the capabilities to be offered in that VRE, i.e. the processes the user will be able to invoke with that authorization.

Further details and information can be found in the gCube Wiki pages related to the service at the following Web address: https://wiki.gcube-system.org/gcube/Data_Mining_Facilities

1.5 STATISTICAL MANAGER ALGORITHMS IMPORTER SERVICE

[4] https://gcube.wiki.gcube-system.org/gcube/Authorization_Client_Library

Prototype scripting is the base of most models in data sciences. Scientists making prototype scripts (e.g. using R and Matlab) often need to share results and make their models used also by other scientists on new data. To this aim, one way is to publish scripts as-a-Service, possibly under a recognized standard (e.g. WPS). The Statistical Algorithms Importer (SAI) is an interface that allows scientists to easily and quickly import R scripts onto DataMiner. DataMiner in turn publishes these scripts as-a-Service and manages multi-tenancy and concurrency. Additionally, it allows scientists to update their scripts without following long software re-deploying procedures each time. In summary, SAI produces processes that run on the DataMiner system and are accessible via the WPS standard.



Main: CMSY_preparation.R

```
require("animation")
#version for NOAA DLM tool
header<-c("Region","Subregion","Stock","Group","Name","EnglishName","ScientificName","Source","MinOfYear","MaxOfYear","StartYear","EndYear","Flim","Fpa","Blim","Bpa","Bmsy","FMSY","MSY","MSYBtrigger

Region<-"Mediterranean"
Subregion<-"Adriatic Sea"
Stock<-"Athe_boy_AD"
Group<-"Plankton feeders"
Name<-"Sand smelt in Adriatic Sea"
EnglishName<-"Big scale sand smelt"
ScientificName<-"Atherina boyeri"
Source<-"-"
MinOfYear<-"1970"
MaxOfYear<-"2014"
StartYear<-"1970"
EndYear<-"2014"
Flim<-"NA"
Fpa<-"NA"
Blim<-"NA"
Bpa<-"NA"
Bmsy<-"NA"
FMSY<-"NA"
MSY<-"NA"
MSYBtrigger<-"NA"
B40<-"NA"
M<-"NA"
Fofl<-"NA"
last_F<-"NA"
Resilience<-"Medium"
r.low<-"NA"
r.hi<-"NA"
stb.low<-"0.2"
stb.hi<-"0.6"
int.yr<-"NA"
intb.low<-"NA"
intb.hi<-"NA"
endb.low<-"0.01"
endb.hi<-"0.4"
q.start<-"NA"
q.end<-"NA"
btype<-"None"
```

Name: CMSY_LEGACY

Description: The CMSY method for data-limited stock assessment. Desc

Category: STOCK_ASSESSMENT

	Name	Created
📁	Backup	03 Dec 08:08 PM 2017
📁	Target	03 Dec 08:08 PM 2017
☐	CMSY.R	03 Dec 07:46 PM 2017
☐	CMSY_preparation.R	03 Dec 07:46 PM 2017
☐	stat_algo.project	03 Dec 07:47 PM 2017

Figure 8. The Statistical Algorithms Importer Graphical User Interface

The SAI interface (Figure 8) resembles the R Studio environment, a popular IDE for R scripts, in order to make it friendly to script providers. The Project button allows creating, opening and saving a working session. A user uploads a set of files and data on the workspace area (lower-right panel). Upload can be done by dragging and dropping local desktop files. As next step, the user indicates the "main script", i.e. the script that will be executed on DataMiner and that will use the other scripts and files. After selecting the main script, the left-side editor panel visualises it with R syntax highlighting and allows modifying it. Afterwards, the user indicates the input and output of the script by highlighting variable definitions in the script and pressing the +Input (or +Output) button: behind the scenes the application parses the script strings and guesses the name, description, default value and type of the variable. This information is visualised in the top-right side Input/Output panel, where the user can modify the guessed information. Alternatively, SAI can automatically fulfil the same information based on WPS4R[5] annotations in the script.

Other tabs in this interface area allow setting global variables and adding metadata to the process. In particular, the Interpreter tab allows indicating the R interpreter version and the packages required by the script and the Info tab allows indicating the name of the algorithm and its description. In the Info tab, the user can also specify the VRE the algorithm should be available to. Once the metadata and the variables information has been fulfilled, the user can create one DataMiner as-a-Service version of the script by pressing the Publish button in the Software panel. The term "software", in this case indicates a Java program that implements an as-a-Service version of the user-provided scripts. The Java software contains instructions to automatically download the scripts and the other required resources on the server that will execute it.

The computations are orchestrated by the DataMiner computing platform that ensures the program has one instance for each request and user. The servers will manage concurrent requests by several users and execute code in a closed sandbox folder, to avoid damage caused by faulty code. Based on the SAI Input/Output definitions written in the generated Java program, DataMiner automatically creates a Web

[5] https://wiki.52north.org/Geostatistics/WPS4R

GUI. During the publication process, the application notifies DataMiner that a new process should be deployed. DataMiner will not own the source code, which is downloaded on-the-fly by the computing machines and deleted after the execution. This approach meets the policy requirements of those users who do not want to share their code. The Repackage button re-creates the software so that the computational platform will be using the new version of the script. The repackaging function allows a user to modify the script and to immediately have the new code running on the computing system. This approach separates the script updating and deployment phases, making the script producer completely independent on e-Infrastructure deployment and maintenance operations. However, deployment is necessary again whenever Input/Output or algorithm's metadata are changed.

To summarise, the SAI Web application relies on the D4Science e-Infrastructure and enables any script, provided by a community of practice working in a VRE, with as-a-Service features. SAI reduces integration time with respect to direct Java code writing. Additionally, it adds (i) multi-tenancy and concurrent access, (ii) scope and access management through Virtual Research Environments, (iii) output storage on a distributed, high-availability file system, (iv) graphical user interface, (v) WPS interface, (vi) data sharing and publication of results, (vii) provenance management and (viii) accounting facilities.

Further details and information can be found in the gCube Wiki pages related to the service at the web addresses: https://wiki.gcube-system.org/gcube/Statistical_Algorithms_Importer

1.6 SPATIAL DATA INFRASTRUCTURE SERVICE (FORMER GIS PUBLISHER SERVICE)

The Spatial Data Infrastructure Service (SDI-Service) is a service acting as a main control point for the entire Spatial Data Infrastructure (SDI). It manages configuration of available GIS Services and their registration into the infrastructure's Information System.

As shown in the figure below, the SDI-Service is designed to interact with the following services:

- **GeoNetwork**: acting as Spatial Data Catalogue;
- **THREDDS, GeoServer**: acting as storage for Spatial Data;
- **GeoFence**: provided by geosolutions (vendor of GeoServer) in order to manage access policies over a cluster of GeoServer.

The main features offered by the SDI-Service are:

- Retrieve SDI Capabilities;
- Translate gCube tokens to credentials on a specific GIS Service instance;
- Manage registration of GIS Services into the Information System, configuring them with required policies;
- Manage dataset collections in GIS Services;
- Publish OGC metadata.

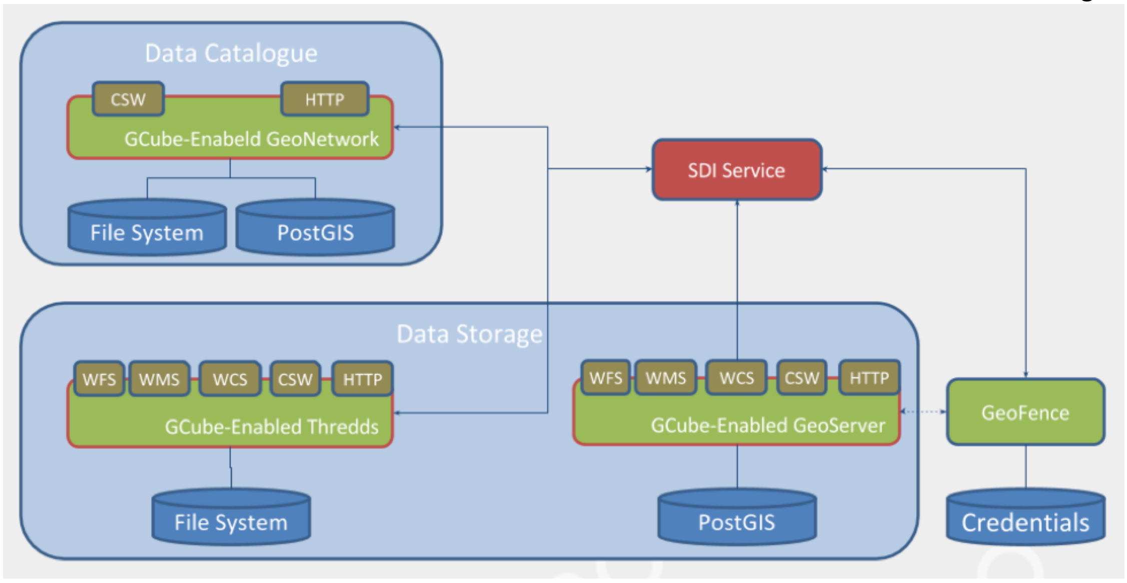


Figure 9. SDI Service Architecture

1.6.1 GCUBE-ENABLED GEO SERVICES

GCube-Enabled GIS Services are service instances which functionalities are extended with components provided by gCube.

As a by-design transparent approach, it:

- Leaves proprietary third-party API and OGC interfaces exposed by the GIS Service intact, allowing the integration of gCube-unaware applications;
- Offers an additional set of operation;
- Allows the integration of gCube Authentication Framework without imposing its adoption to clients.

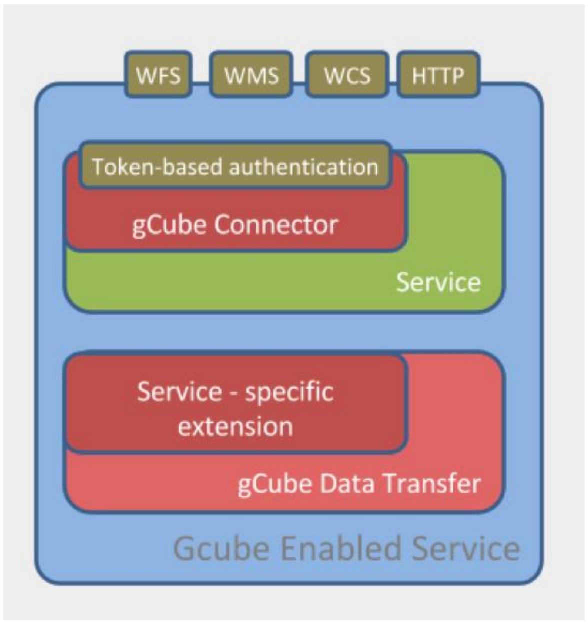


Figure 10. gCube-Enabled Service architecture

As shown in Figure 10, a gCube-Enabled GIS Service is composed of:

- Service: the GIS Service to be enhanced;
- gCube Connector: HTTP filter which intercepts calls to the Service, performing authentication to it based on the specified gCube token (if any);

- gCube Data Transfer: gCube Service which grants controlled CRUD operations on Service's File System;
- Service-specific extension: Set of additional functionalities exposed by gCube Data Transfer.

1.6.2 GIS INTERFACE

GIS-Interface is a Java library which exposes methods to access, modify and publish spatial data and related metadata on a Spatial Data Infrastructure (SDI).

The GIS-Interface library relies on the libraries geoserver-manager and geonetwork-manager provided by GeoSolutions (vendor of GeoNetwork and GeoServer) in order to interact with REST APIs offered by GIS Services.

GIS-Interface offers advanced APIs to manage both data and metadata in atomic operations, allowing also to directly exploit APIs offered by GeoSolutions libraries.

Publishing of GIS metadata is made using GeoNetwork[6] library. The library exposes facilities to help developers define layers metadata compliant with gCube SDI policies.

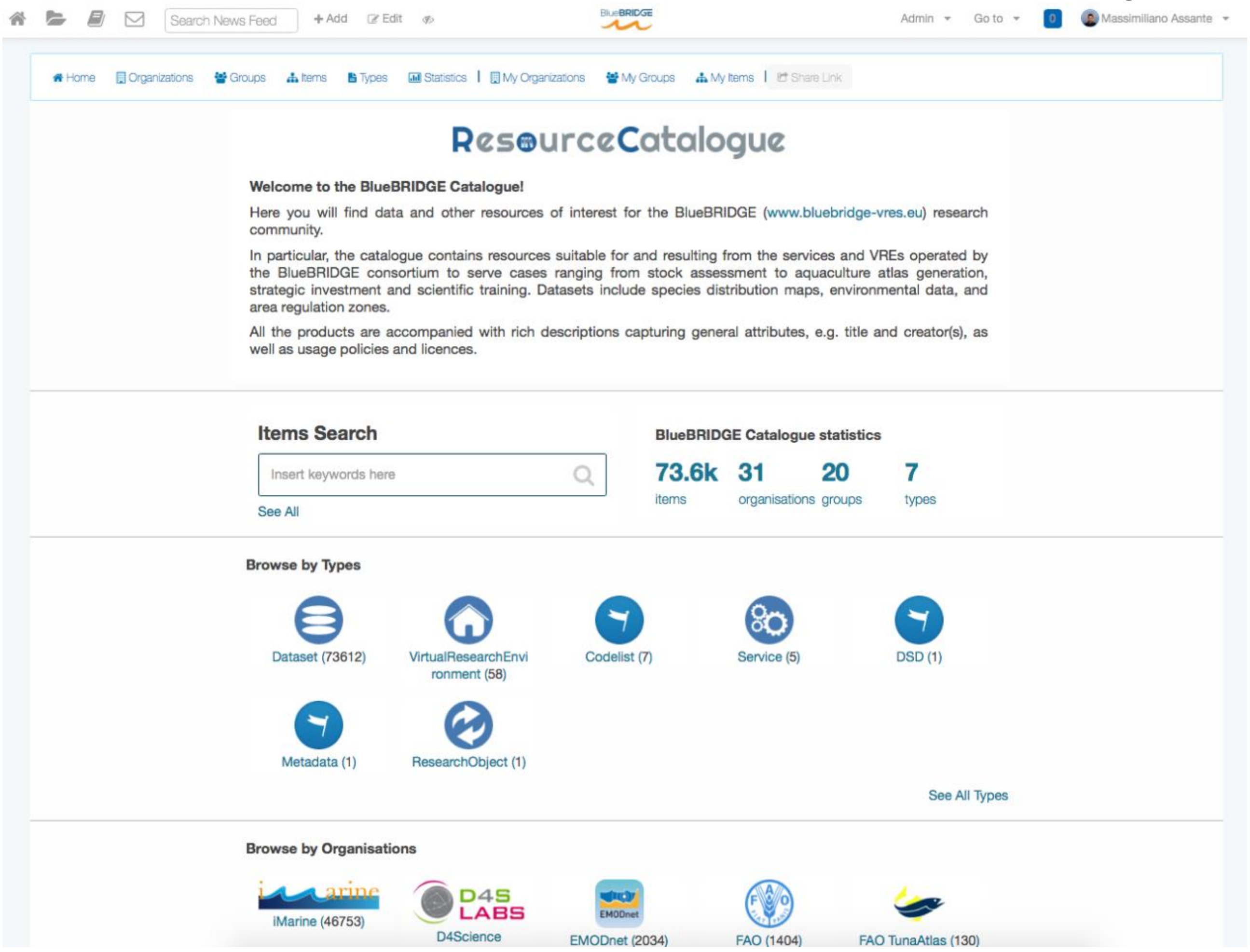
Further details and information can be found in the gCube Wiki page related to the service at the web address: https://wiki.gcube-system.org/gcube/GIS_Interface.

1.7 RESOURCE CATALOGUE SERVICE

The Resource Catalogue (https://bluebridge.d4science.org/catalogue) service is conceived to enable the definition, publishing, discovery and access to resources of various genre (e.g. datasets, records). It contains a wealth of resources resulting from several activities and communities occurring within BlueBRIDGE VREs. All the products can be accompanied with rich descriptions capturing general attributes, e.g. title and creator(s), as well as usage policies and licences. The Catalogue service is built upon the CKAN platform[7], which offers a rich set of facilities on its own. In fact, CKAN is a powerful Data Management System (DMS) that makes data accessible by providing tools to streamline publishing, sharing, finding and using data. CKAN is an open-source DMS for powering data hubs and data portals. It makes easy to publish, share and use data.

[6] https://wiki.gcube-system.org/gcube/GeoNetwork_library

[7] CKAN Platform: https://ckan.org



Welcome to the BlueBRIDGE Catalogue!

Here you will find data and other resources of interest for the BlueBRIDGE (www.bluebridge-vres.eu) research community.

In particular, the catalogue contains resources suitable for and resulting from the services and VREs operated by the BlueBRIDGE consortium to serve cases ranging from stock assessment to aquaculture atlas generation, strategic investment and scientific training. Datasets include species distribution maps, environmental data, and area regulation zones.

All the products are accompanied with rich descriptions capturing general attributes, e.g. title and creator(s), as well as usage policies and licences.

Items Search

Insert keywords here

See All

BlueBRIDGE Catalogue statistics

73.6k	31	20	7
items	organisations	groups	types

Browse by Types

Dataset (73612)
VirtualResearchEnvironment (58)
Codelist (7)
Service (5)
DSD (1)
Metadata (1)
ResearchObject (1)

See All Types

Browse by Organisations

iMarine (46753)
D4Science
EMODnet (2034)
FAO (1404)
FAO TunaAtlas (130)

Figure 11. The BlueBRIDGE Resource Catalogue service GUI

Although the Catalogue service is built upon the CKAN platform, several modifications/enhancements have been added to it.

Some of these modifications concern the service architecture. In particular, it has been enriched with (i) multi-tenancy capabilities (tenants may be given the ability to customize some parts of the application, such as business rules and colour of the GUI), (ii) the possibility to potentially equip each VRE with a "tailored" dedicated catalogue, i.e. a catalogue instance specifically configured to meet the requirements of a VRE in terms of metadata formats, harvested data sources etc. yet capable of being harvested from the BlueBRIDGE global resource catalogue, and, (iii) the capability of being seamlessly connected to the BlueBRIDGE Gateway (which is the main access point of the BlueBRIDGE Infrastructure), i.e. single sign-on and fully integrated (roles and permission) catalogue .

Further modifications concerned the supported data model. The concepts of *gCube Metadata profile*[8] (defining a Metadata schema XML-based for adding custom metadata fields) and *Product Type* (configurable Type based on the custom metadata format defined on a given catalogue instance) have been implemented. Additionally, the GUI of the Catalogue service has been completely redesigned (see Figure above) to facilitate and enhance the usability and better user experience.

[8] https://wiki.gcube-system.org/gcube/GCube_Data_Catalogue#gCube_Metadata_Profile

To facilitate the integration of the catalogue with existing data sources, including other catalogues, some plug-ins have been developed. In particular, the plugin for harvesting metadata from geo-referenced metadata catalogues via the OGC CSW[9] protocol, e.g. GeoNetwork catalogues, has been rewritten.

1.7.1 Publishing Products in the Resource Catalogue

The publishing of Catalogue Products in the catalogue has been improved with respect to the standard CKAN capability. Specifically, entitled users can explicitly publish new resources by using three different modalities: (i) via workspace, (ii) via custom portlet, and (iii) via a dedicated web service exposing a REST API.

Figure 12 depicts the menu item options added to the workspace service to enable authorised users to publish a single file or a set of files (stored in a folder) into the catalogue. When clicking on the “Publish on Catalogue” entry the user is requested (a) to provide information on the resources he/she is going to publish (information depends on the selected resource typology) and (b) to select the workspace objects to associate with the resource (if multiple files are there).

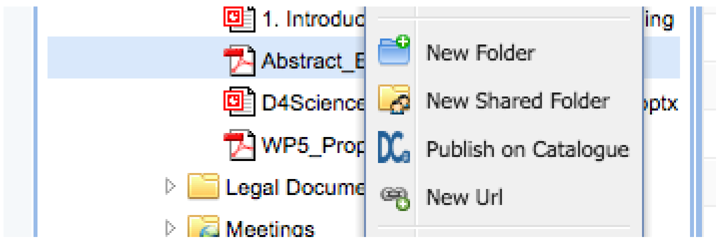
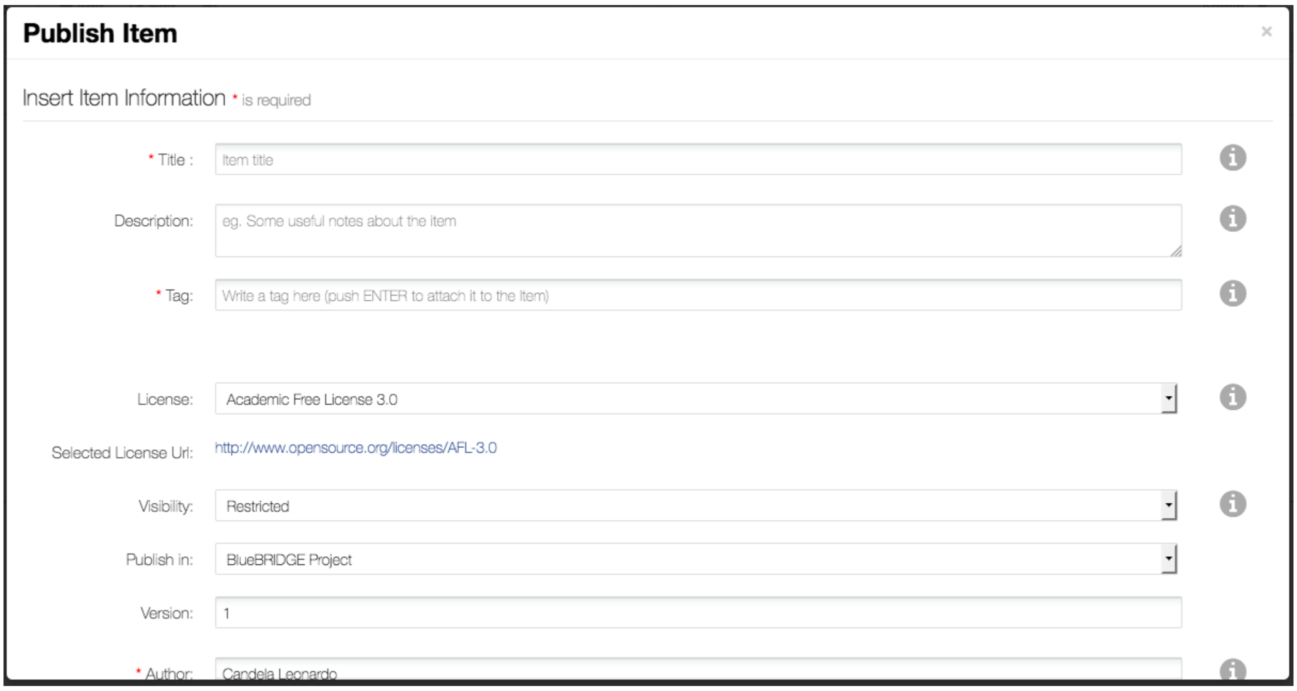


Figure 12. Resource Catalogue: Publish via Workspace Option

Figure 13 depicts the Publish Item portlet developed to support the publishing of resources into the catalogue. When clicking on the “Publish Item” entry of the Resource Catalogue GUI, like in the Workspace scenario just described, the user is requested to (a) to provide information on the resources he/she is going to publish and (b) to select the objects to associate with the resource.



Publish Item

Insert Item Information * is required

* Title :	Item title
Description:	eg. Some useful notes about the item
* Tag:	Write a tag here (push ENTER to attach it to the item)
License:	Academic Free License 3.0
Selected License Url:	http://www.opensource.org/licenses/AFL-3.0
Visibility:	Restricted
Publish in:	BlueBRIDGE Project
Version:	1
* Author:	Candela Leonardo

Figure 13. Resource Catalogue: Publish Widget

[9] https://en.wikipedia.org/wiki/Catalog_Service_for_the_Web

The third option to publish resources into the Catalogue is based on a REST-based service and documented in the following section.

1.7.2 Resource Catalogue Web Service: programmatic access to the Resource Catalogue

A complete and detailed description of the Resource Catalogue Web Service is available by a gCube Wiki dedicated page[10].

The web service is envisaged to support operation on the following entities/catalogue concepts:

- *Organisations*, i.e. the contexts in which publishing happen. This notion is paired with the VRE context, i.e. organisations corresponds to VRE and once publishing in a VRE the resource is associated with the homonymous organisation. The web service enacts to create, update, delete, and retrieve organisations;
- *Groups*, i.e. a collection of resources. This notion is not automatically paired with other concepts and clients can rely on them to build custom collections of published items for simplifying discovery. The web service enacts to create, update, delete, and retrieve groups;
- *Items*, i.e. the catalogue items corresponding to the resources. The web service enacts to create, update, delete, and retrieve items;
- *Resources*, i.e. the actual content of a published resource (usually a file or another entity accessible by an URL). The web service enacts to create, update, delete, and retrieve resources associated with an item;
- *Item profiles*, i.e. the information schema characterising a resource typology (the metadata profile). The web service enacts to create, update, delete, and retrieve metadata profiles.

1.7.3 Enacting Semantic Data Discovery on Resource Catalogue

The Resource Catalogue offers seamless access to a wide spectrum of resources. The actual resources are stored along with a set of information (metadata) that allow them to be easily discovered. In order to enhance the discovery of resources that exist in the catalog we exploit a knowledge base with semantic descriptions of the resources. The knowledge base resembles a semantic graph which can be queried using SPARQL queries. In the sequel we will call this knowledge base as semantic catalog.

The semantic catalogue is being constructed using the contents that have already been published in the resource catalogue. The metadata of these resources are exported using the appropriate CKAN plugins. In particular the ckanext-dcat[11] plugin is used which exposes the contents of a CKAN catalog as an RDF document, using DCAT[12] ontology. The following figure shows the architecture of the components that are responsible for constructing and exposing the contents the semantic catalog.

The following diagram shows the components that are used for making the semantic data discovery functional. More specifically:

- The Resource Catalog, containing the actual resources. The contents are exposed by a set of services and a graphical user interface;
- The Knowledge Base, containing the RDF descriptions of the resource catalog contents. The knowledge base can be queried using RDF standards (i.e. SPARQL);
- The Resource Catalog Harvester, which is responsible for harvesting all the data from the resource catalog and import them as RDF data in the knowledge base.

[10] https://wiki.gcube-system.org/index.php?title=Catalogue_restful_service
[11] https://github.com/ckan/ckanext-dcat
[12] https://www.w3.org/TR/vocab-dcat/

Apart from the basic metadata (i.e. the title, the owner, the modification date, etc.) that are already exposed, the plugin is configurable in terms of the metadata that will be further exposed (using profiles). This gives the flexibility of defining custom mappings between CKAN metadata fields and RDF descriptions and therefore exposing the resource catalog contents using rich descriptions.

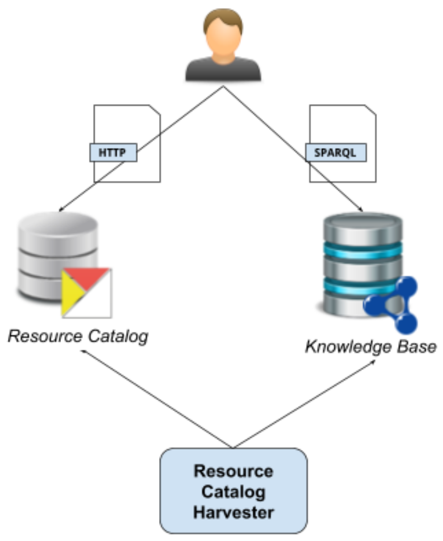


Figure 14. Components used for making the semantic data discovery functional

The benefits gained from the knowledge base are many. Just indicatively, its resources are identified using URIs, and therefore they can be linked with other resources. This allows the enrichment of resources with information that can be found in public repositories (e.g. dbpedia[13]). In addition, since the data can be queried using standard RDF technologies (i.e. SPARQL), it can be used for answering complex queries that cannot be answered directly from the resource catalog. For example, the queries "Find all the resources, that have the type Dataset, and have been published by a particular user, within 2016 and 2017, and are published using an open license" or "Find all the resources, that have a type GRSF[14] record, that describe a stock record with recent time-series (after 2016)" can be translated into SPARQL queries that can be answered directly from the knowledge base. Of course, the same queries can be answered from the resource catalog, however this requires interaction with a human (first search using generic terms, then click on particular tags to restrict the answer, etc.). Furthermore, the knowledge base can be discovered using alternatives approaches, like SPARKLIS[15], PascoLink[16], etc.

Further details and information can be found in the gCube Wiki page related to the Resource Catalogue service at the web address: https://wiki.gcube-system.org/gcube/GCube_Resource_Catalogue.

[13] http://wiki.dbpedia.org
[14] https://i-marine.d4science.org/web/grsf
[15] http://www.irisa.fr/LIS/ferre/sparklis/
[16] http://www.ics.forth.gr/isl/PascoLink

2 Facilities for VRE Management and Usage

2.1 VRE Management Facilities

VRE Management facilities comprises a set of services and applications offering functions for defining, creating and deploying VREs. These services support VRE Designers and Managers through graphical user interfaces to instruct the infrastructure about the expected features of the desired VRE as well as allowing to easily update the VRE once defined and operational.

A Virtual Research Environment (VRE) is an application with the following characteristics: (i) it is a Web-based working environment; (ii) it is tailored to serve the needs of a Community of Practice; (iii) it is expected to provide a community of practice with the whole array of commodities needed to accomplish the community's goal(s); (iv) it is open and flexible with respect to the overall service offering and lifetime; and (v) it promotes fine-grained controlled sharing of both intermediate and final research results by guaranteeing ownership, provenance, and attribution.

The administration of these cooperation environments is a four-tasks activity envisaging:

- a definition phase in which a user having the role of VRE Designer specifies the characteristics of a new VRE conceived to serve an application scenario;
- an **approval phase** in which the VRE Manager decides whether the specified VREs ca be accepted or rejected. For what is concerned with the accepted VRE, the VRE Manager decides also how this VRE has to be deployed, e.g. which hosting nodes will be exploited;
- a **verification phase** in which the VRE Manager validates a VRE resulting from the approval phase;
- a **management phase** in which the VRE Manager operates on a deployed VRE in order to customise specific aspects (e.g. the layout governing the placement of user interfaces constituents a.k.a. portlets) and monitors the operational state of the VRE as a whole.

Further details and information can be found in the gCube Wiki page related to the service at the web address: https://wiki.gcube-system.org/gcube/VRE_Administration.
This services pre-exists BlueBRIDGE, during the period they have been re-designed to adapt to the evolving technologies.

2.2 VRE Enabling Portlets

A set of interaction-oriented services and a front-end applications providing functions to manage VRE Users and to support them to collaborate, cooperate and exchange content/information by using social media-like tools.

Although these front-end applications (portlets) pre-exist BlueBRIDGE, during the project period they have been re-designed to adapt to the evolving technologies, in particular they are now responsive (capable of displaying on different devices such as smartphones, tablets and desktop).

2.2.1 Social Networking Portlets

Through Social Networking Portlets, users can share posts in VREs, attach files to these posts, view, comment, like and unlike them by means of the Share-Update and News-Feed web applications. These applications mainly relies on the facilities offered by the Social-Networking library. Both applications are responsive, thanks to the Bootstrap[17] framework, so that users can share post or view it by means of tablets or smartphones too.

2.2.1.1 Share Updates portlet

The Share-Update application allows users to share posts within a VRE context. It shows the current user's avatar on the left, a textbox area to write, a select box to choose if other vre members should be notified about the post and, on the right side, the file upload and share buttons. If the user wishes to notify other members for his post, she can do that by selecting the proper option within the selection box.
The portlet allows also to attach a single file or more than one file by means of an upload file by means of drag & drop. If an URL is pasted into the textbox area, the application automatically tries to generate a "link preview" to be attached to the post. Further social networking tools such as hashtags for topics (using '#' notation), to cite other people or groups (using '@' notation) are supported.

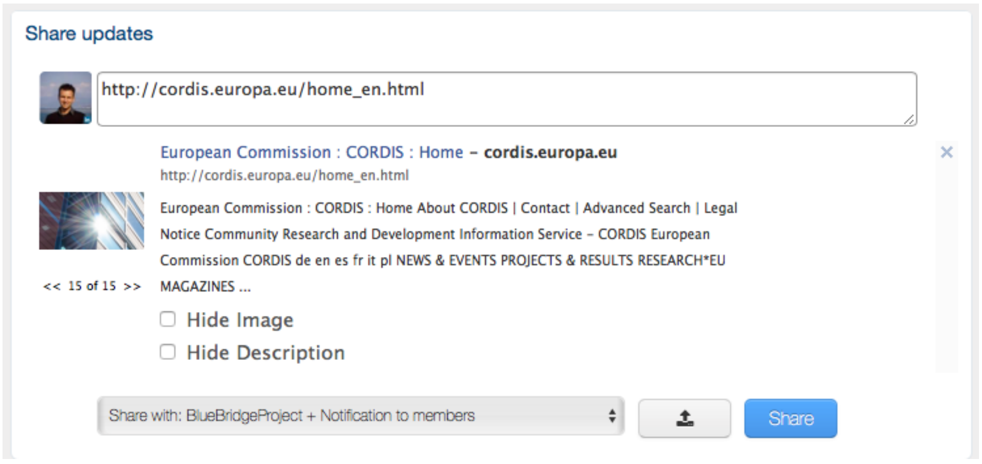

Share updates
http://cordis.europa.eu/home_en.html
European Commission : CORDIS : Home – cordis.europa.eu
http://cordis.europa.eu/home_en.html
European Commission : CORDIS : Home About CORDIS | Contact | Advanced Search | Legal Notice Community Research and Development Information Service – CORDIS European Commission CORDIS de en es fr it pl NEWS & EVENTS PROJECTS & RESULTS RESEARCH*EU MAGAZINES ...
<< 15 of 15 >>
Hide Image
Hide Description
Share with: BlueBridgeProject + Notification to members
Share

Figure 15. Share Updates portlet with link preview

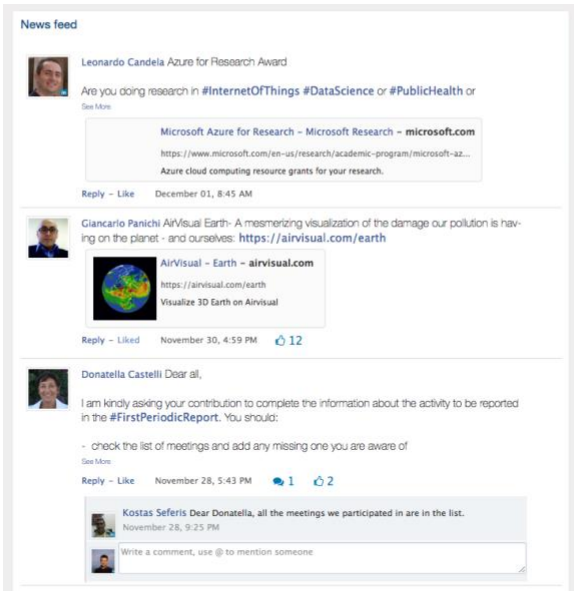
Further details and information can be found in the gCube Wiki page related to the service at the web address: https://wiki.gcube-system.org/gcube/Sharing_Posts_and_using_News_feed - Share_Updates.

2.2.1.2 News Feed Portlet

The News-Feed portlet allows to view already published posts sorted in a reverse chronological order within a given VRE context, along with their comments and likes, plus some other contextual information such as the date in which the post was published, the list of attached files, etc. Moreover, it allows to add new comments to the post and to like or unlike it.

The portlet is fully integrated with the Social Networking Data Discovery service (see Sec. 1.3.2) so, when a user searches for something, the list of retrieved posts is automatically shown into the application. The same happens when the user is interested in viewing his own statistics (e.g. such as the posts he has recently wrote) and when the user searches posts that contain a given topic (e.g. '#gcube').

[17] https://getbootstrap.com/


News feed

Leonardo Candela Azure for Research Award

Are you doing research in #InternetOfThings #DataScience or #PublicHealth or
See More

Microsoft Azure for Research – Microsoft Research – microsoft.com
https://www.microsoft.com/en-us/research/academic-program/microsoft-az...
Azure cloud computing resource grants for your research.

Reply – Like December 01, 8:45 AM

Giancarlo Panichi AirVisual Earth- A mesmerizing visualization of the damage our pollution is having on the planet - and ourselves: https://airvisual.com/earth

AirVisual – Earth – airvisual.com
https://airvisual.com/earth
Visualize 3D Earth on Airvisual

Reply – Liked November 30, 4:59 PM 12

Donatella Castelli Dear all,

I am kindly asking your contribution to complete the information about the activity to be reported in the #FirstPeriodicReport. You should:

- check the list of meetings and add any missing one you are aware of
See More

Reply – Like November 28, 5:43 PM 1 2

Kostas Seferis Dear Donatella, all the meetings we participated in are in the list.
November 28, 9:25 PM

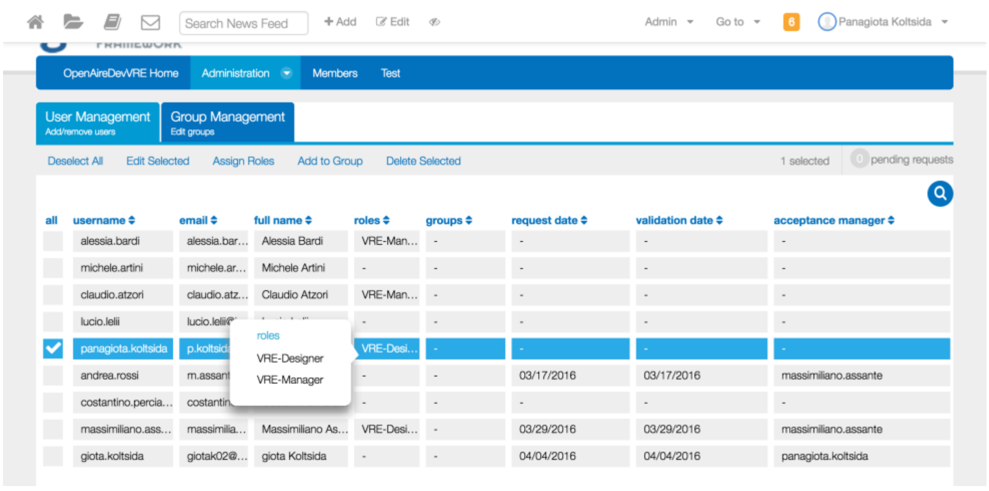
Write a comment, use @ to mention someone

Figure 16. News Feed portlet showing 3 posts along with comments and likes

Further details and information can be found in the gCube Wiki page related to the service at the web address: [https://wiki.gcube-system.org/gcube/Sharing_Posts_and_using_News_feed#News_Feed.](https://wiki.gcube-system.org/gcube/Sharing_Posts_and_using_News_feed#News_Feed)

2.2.2 Users and Groups Management Portlet

The Users and Groups Management Portlet provides functionalities for managing users and groups for each VO/VRE. This portlet can only be accessed by Administrators. The portlet is divided into 2 main tabs offering distinct functionality for the management of users and groups respectively.



all	username	email	full name	roles	groups	request date	validation date	acceptance manager
	alessia.bardi	alessia.bar…	Alessia Bardi	VRE-Man…	-	-	-	-
	michele.artini	michele.ar…	Michele Artini	-	-	-	-	-
	claudio.atzori	claudio.atz…	Claudio Atzori	VRE-Man…	-	-	-	-
	lucio.lelii	lucio.lelii@		-	-	-	-	-
✓	panagiota.koltsida	p.koltsid		VRE-Desi…	-	-	-	-
	andrea.rossi	m.assant		-	-	03/17/2016	03/17/2016	massimiliano.assante
	costantino.percia…	costantin		-	-	-	-	-
	massimiliano.ass…	massimilia…	Massimiliano As…	VRE-Desi…	-	03/29/2016	03/29/2016	massimiliano.assante
	giota.koltsida	giotak02@…	giota Koltsida	-	-	04/04/2016	04/04/2016	panagiota.koltsida

Figure 17. Users and Groups Management Portlet showing 2 tabs for users and groups

Further details and information can be found in the gCube Wiki page related to the portlet at the web address: [https://wiki.gcube-system.org/gcube/Users%27_Management.](https://wiki.gcube-system.org/gcube/Users%27_Management)

3 SOFTWARE DISTRIBUTION

All the software produced for the gCube system follows a common integration, testing and release process and, therefore, all facilities share the same location for:

- **source code**: available on GitHub at https://github.com/gcube-system/gcube-releases;
- **binary packages**: available in the gCube download page on the gCube website at https://www.gcube-system.org/software-releases.

All the facilities described in this document are delivered with the EUPL v1.1 software license.

REFERENCES

[1] BlueBRIDGE project wiki - https://support.d4science.org/projects/bluebridge/wiki

[2] gCube wiki - https://wiki.gcube-system.org/gcube/About_gCube

[3] Apache Cassandra - https://cassandra.apache.org/

[4] Elastic Search - https://www.elastic.co/products/elasticsearch

[5] MongoDB - https://www.mongodb.com/

[6] Apache JackRabbit - http://jackrabbit.apache.org/jcr/index.html

[7] Coro, G., Panichi, G., Scarponi, P., & Pagano, P. (2017). Cloud computing in a distributed e-infrastructure using the web processing service standard. *Concurrency and Computation: Practice and Experience*, *29*(18) 10.1002/cpe.4219

[8] Schut P, Whiteside A. OpenGIS Web Processing Service 2007. OGC project document http://www.opengeospatial.org/standards/wps

[9] Lebo T, Sahoo S, McGuinness D, Belhajjame K, Cheney J, Corsar D, Garijo D, Soiland-Reyes S, Zednik S, Zhao J. Prov-o: The prov ontology. W3C Recommendation 2013; 30.

[10]European Grid Infrastructure. The European Grid Infrastructure Federated Cloud 2016. www.egi.eu/federation/

[11]Candela L, Castelli D, Coro G, Pagano P, Sinibaldi F. (2013) Species distribution modeling in the cloud. Concurrency and Computation: Practice and Experience, 28(4) 10.1002/cpe.3030

[12]Berghe EV, Coro G, Bailly N, Fiorellato F, Aldemita C, Ellenbroek A, Pagano P. (2015) Retrieving taxa names from large biodiversity data collections using a flexible matching workflow. Ecological Informatics 2015; 28:29–41 10.1016/j.ecoinf.2015.05.004

[13]Assante M, Candela L, Cirillo R, Coro G, Koltsida P, Marioli V, Perciante C, Sinibaldi F, Pagano P. (2016) BlueBRIDGE VRE Commons Facilities. BlueBRIDGE Deliverable D9.1