iMarine Project

Activities Report

Department: Information Technology
Group: Support for Distributed Computing

Name: Nikolaos Drakopoulos
Email: nickdrakop@gmail.com, sdi0600009@di.uoa.gr
Activities:

- T5.3 Monitoring and Accounting
- T9.2 Data Transfer facilities
- T11.2 Data Management APIs

Supervisor: Andrea Manzi

Table of contents:

Table of figures:

Introduction

I worked for CERN at the Information Technology Department (Support for Distributed Computing) in the iMarine project from July 2012 to June 2013.
iMarine project is a data infrastructure initiative to support the Ecosystem Approach to fisheries management and conservation of marine living resources. It constitutes a continuation of the DILIGENT and D4Science projects.
My activities relied on implementing and integrating a set of facilities for providing a data transfer mechanism between the nodes of the D4Science Ecosystem.

Technology

The D4Science infrastructure is developed and operated by using the gCube technology.

gCube is the Software System designed and implemented to enable the building and operation of a Service Oriented Infrastructure supporting the definition of Virtual Research Environments (VREs).

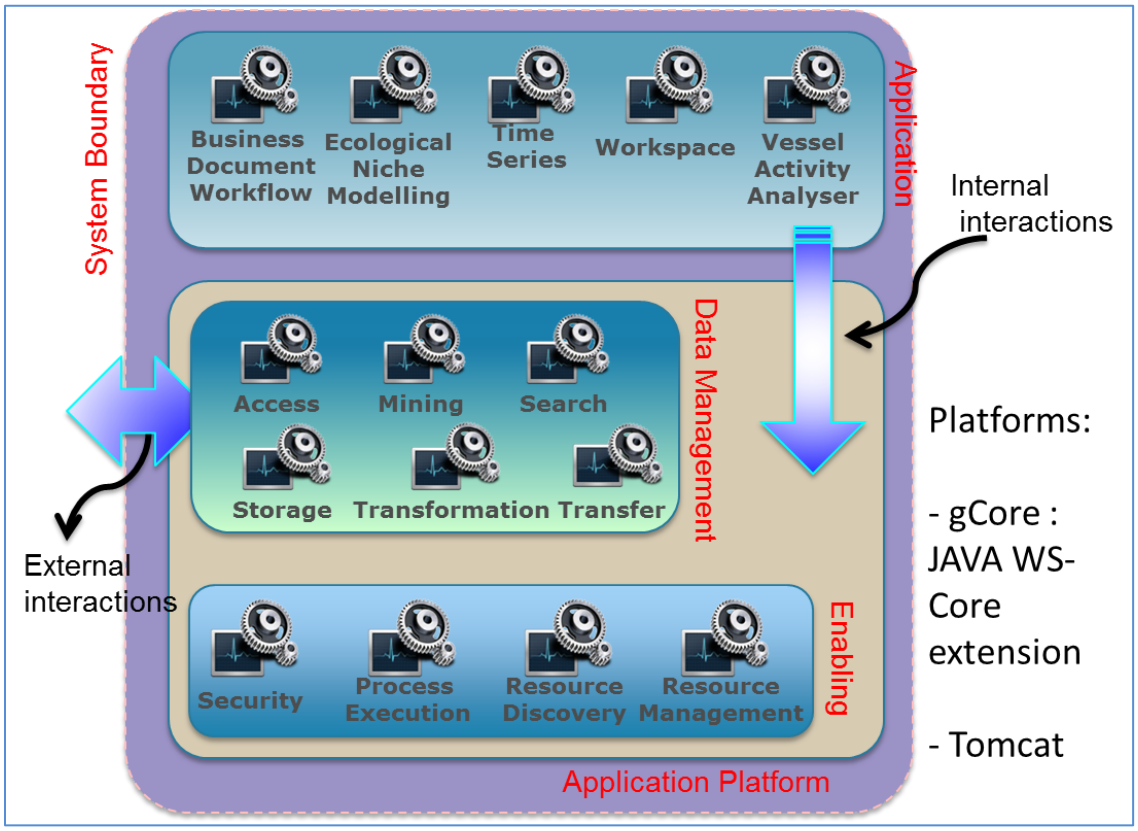


Figure 1: gCube Framework

gCube Hosting Node (gHN) is the topological unit of a gCube infrastructure. It constitutes an abstraction over a container running on a given port and hosting at least a minimal set of basic gCube services (the local services) dedicated to the host management.

1. Data Transfer

The data transfer section groups the activities for Data Transfer Facilities (T9.2) and Data Management API (T11.2).

1.1. Philosophy

Data transfer on a distributed infrastructure has to guarantee in first place transfer reliability and optimization in the sense of the resource usage (minimize network load while not causing storage overload). In addition compared to most of the solution developed for data transfer, the solution designed has to take into account not only the standard "unstructured" data transfer (file transfer) but the capability of "structured" data transfer peculiar to the iMarine data infrastructure.

1.2. Architecture

The main components forming the class of Data transfer facilities are:

- The Data Transfer Scheduler service (gDT Scheduler)
- The Data Transfer Scheduler Library (gDT Scheduler Lib)
- The Data Transfer Agent service (gDT Agent)
- The Data Transfer Agent Library (gDT Agent Lib)
- The Data Transfer Portlet (gDT Portlet)

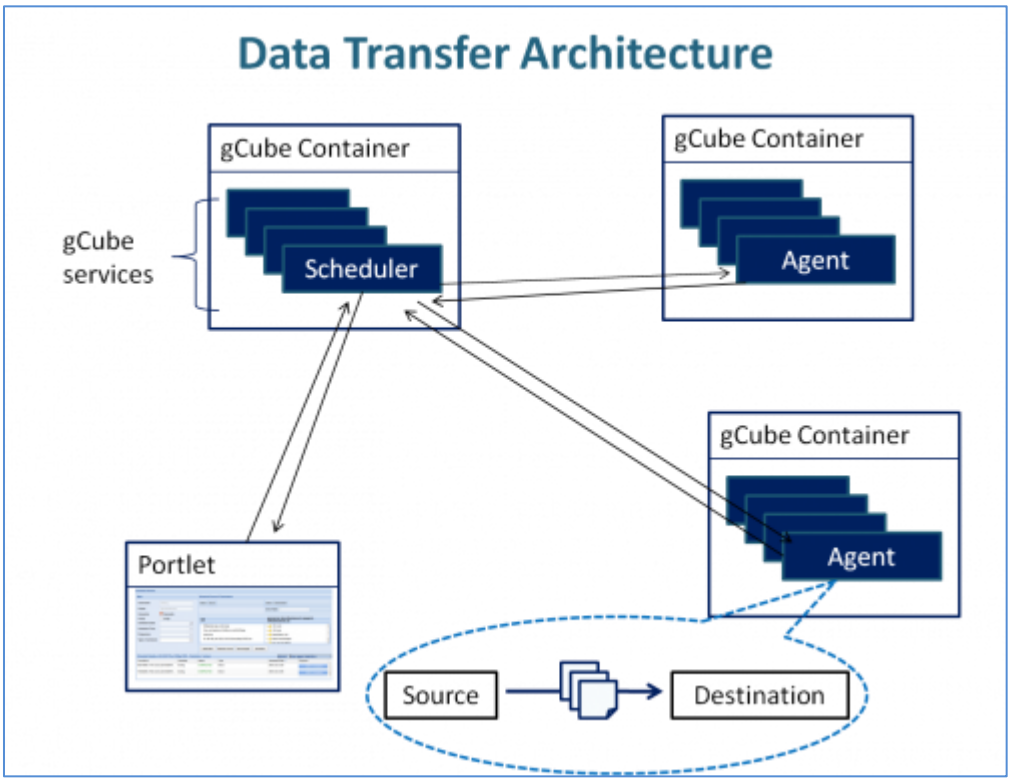


Figure 2: Data Transfer Architecture

gCube Data Transfer Components:
https://gcube.wiki.gcube-system.org/gcube/index.php/GCube_Data_Transfer_Facilities

1.3. gDT Scheduler Service & Library

The Data Transfer Scheduler Service is responsible for the transfer scheduling activity delegating and spawning the transfer logic to the series of gDT Agent deployed on the infrastructure. It relies on Messaging to consume transfer results from gDT Agent.

1.3.1. Types of Transfer

The client can set a transfer between these types:

- File-Based Transfer (unstructured)

The file-based Transfer includes transfer cases **from** *Workspace/ MongoDB/ DataSource/ Agent's node/ URI* ***to*** *MongoDB/ DataStorage/ Agent's node.*

The Workspace represents the collaborative area where users can exchange and organize information objects (items) according to their specific needs in the iMarine gateway. Every user of any Virtual Research Environment is provided with this area for the exchange of workspace objects with other users.

The *DataSource* and *DataStorage* are specified in the Information system with an XML schema. They constitute several servers of the infrastructure that can be accessed via HTTP/FTP/WebDAV and transfer files to/from.

The *Agent's node* is a node that runs the Agent Service and gives access to a specific folder of its file system. This folder can be specified in the configuration files of the agent's service.

- Tree-Based Transfer (structured)

The tree-based Transfer includes the transfer case of transferring from a tree collection to another one by using the Tree Manager Framework.

The tree-based access/transfer is part of the Data Access and Storage Facilities. A cluster of components within the system focuses on uniform access and storage of structured data of arbitrary semantics, origin, and size. Some additional components build on the generic interface and specialize to data with domain-specific semantics, including document data and semantic data. Access and storage of structured data can be provided under a uniform model of labeled trees and through a remote API of read and write operations over such trees. A tree-oriented interface is ideally suited to clients that abstract over the domain semantics of the data. The tree interface is collectively provided by a heterogeneous set of components such as the Tree Manager Framework.

1.3.2. Types of Schedule

There are three types of schedule:

- Direct Transfer
- Manually Scheduled Transfer
- Periodically Scheduled Transfer

In the Direct Transfer the client can simply submit a transfer without any schedule. The service directly through the agent library performs the transfer.

In the Manually Scheduled Transfer the client submit a transfer by providing also the date that he wants to start the transfer. More specifically the given date should be only a specific instance and the transfer will take place only once.

The actual schedule exists in the Periodically Scheduled Transfer where the client sets the period that he wants the transfer to take place. He can choose one of the six given options: every *minute/ hour/ day/ week/ month/ year*. At this type of schedule, the client should also give the start date of the schedule. This is a specific instance like in the manually scheduled transfer and constitutes the beginning of the scheduled transfer.

1.3.3. Transfer Status

The several status points are:

- *STANDBY (the transfer has not started yet)*
- *ONGOING*
- *CANCELED*
- *FAILED*
- *COMPLETED*

1.3.4. Scheduler Library

The Data Transfer Scheduler Library is the Client Library implementing the API for Data Transfer Scheduling. In particular it consists of two separate API's, one for the management and one for the scheduling.

The management API is responsible for:

- Retrieving information about the transfers.
- Retrieving objects from the IS (agents, data sources etc.).
- Check the existence of the above objects in DB.
- Retrieving information regarding the agent statistics.

The scheduling API is responsible for submitting a specific or several transfer operations, cancel a transfer, monitor a transfer and getting the outcomes of a transfer.

Wiki page:
https://gcube.wiki.gcube-system.org/gcube/index.php/Data_Transfer_Scheduler

1.4. Scheduler DB Interface

The Data Transfer Scheduler Database Library implements the API so that the Data Transfer Scheduler Service (either the scheduler one or the management) can access the database.

The main structure of the DB can be shown at the following figure.

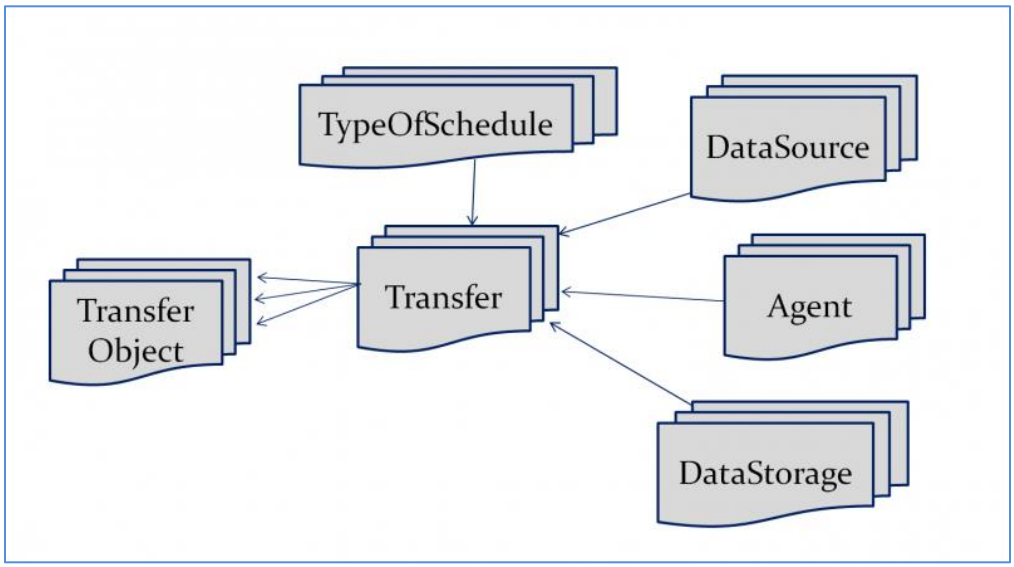


Figure 3: The Main Structure of DB

Wiki page:
https://gcube.wiki.gcube-system.org/gcube/index.php/Data_Transfer_Scheduler#Data_Transfer_Scheduler_Database_Library

1.5. Scheduler IS Interface

The Data Transfer Scheduler IS Library implements the API so that the Data Transfer Scheduler Service can retrieve needed info about the Information System and store them in the Database. It also uses the DB interface in order to access the DB of the scheduler service.

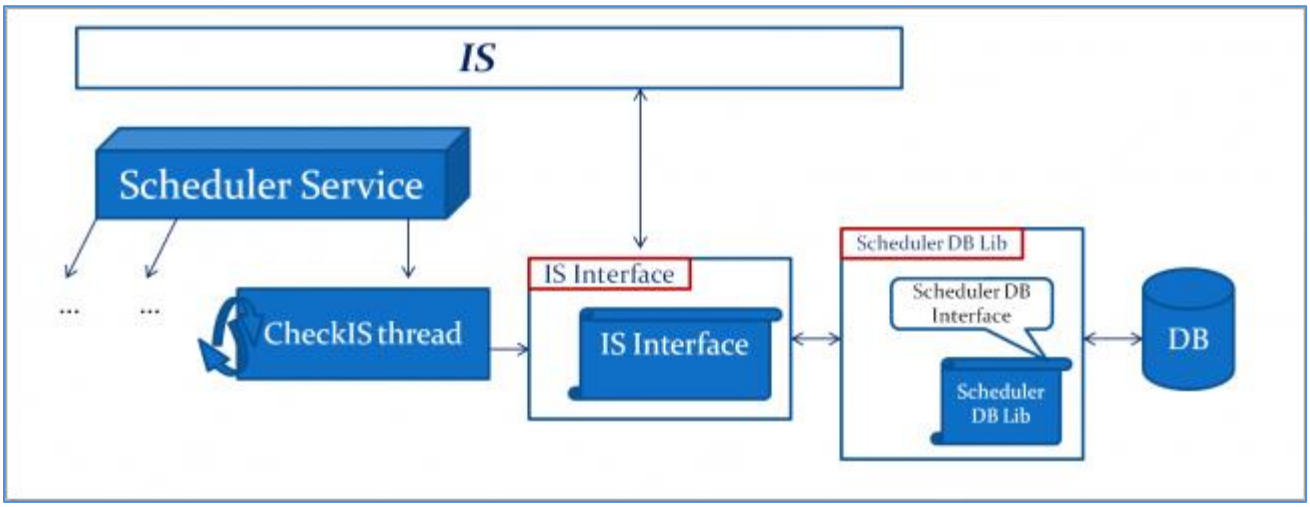


Figure 4: The IS Interface for the scheduler

Wiki page:
https://gcube.wiki.gcube-system.org/gcube/index.php/Data_Transfer_Scheduler#Data_Transfer_Scheduler_IS_Library

1.6. Common Messaging

The communication between scheduler service and agent service is being done via Common Messaging Interface. This interface provides the messaging system so that the scheduler service can produce messages regarding a transfer performance and the agent can consume these messages on the other side. The other way around is for the agent to be able to produce the transfer results and the scheduler consume them.

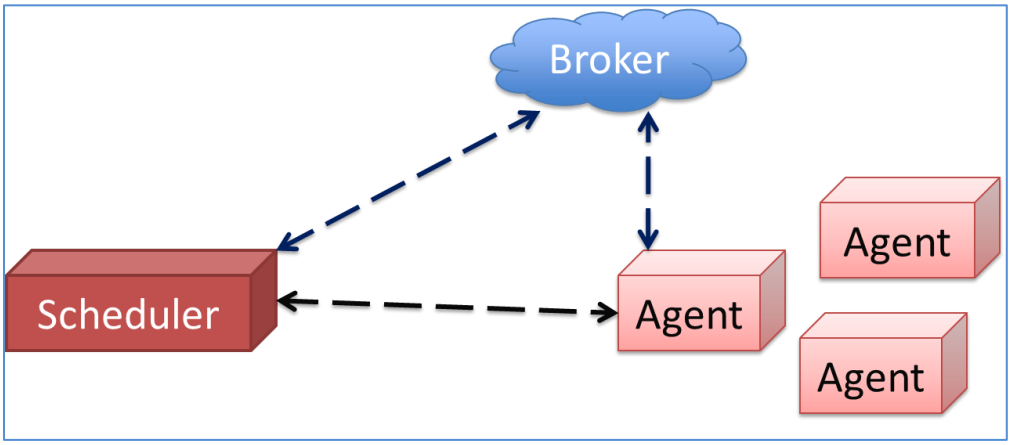


Figure 5: Common Messaging Interface

Wiki page:
https://gcube.wiki.gcube-system.org/gcube/index.php/Data_Transfer_Common_Messaging

1.7. gDT Agent Service & Library

The Data Transfer Agent Service has been implemented with the aim of facilitating the transfer of data (both structured and unstructured) for the following use cases:

- Transfer of Local files from an external/internal client to a remote GHN
- File Transfer from a remote Data Source to a remote GHN using standard protocol (HTTP, FTP, WebDAV, etc.) by integrating the Apache VFS Library.
- File Transfer from a remote Data Source to the gCube Storage Manager (MongoDB) using standard protocol (HTTP, FTP, WebDAV) by integrating the Storage Manager Library and the Apache VFS Library.
- File Transfer from a remote Data Source to a remote Data Storage.
- Tree based data transfer from a tree-collection to another.

In addition the service exploits the Messaging infrastructure in order to publish transfer statistics that can be consumed by:

- Accounting statistics consumers
- The Data Transfer Scheduler Service, which use messaging in order to consume Agent transfer results.

gCube Storage Manager is a library for access and storage of unstructured byte streams, or file in the infrastructure's storage repository (MongoDB).
The library acts a facade to the service and allows clients to download, upload, remove, add, and list files. Files have owners and owners may define access rights to files, allowing private, public, or group-based access.

1.7.1. Transfer Status

When using the Asynchronous transfers facilities, each submitted transfer can be monitored using the MonitorTransfer method. The output of the operation is a TransferStatus enum value (defined on the Data Transfer Library).
The several status points are: *QUUED, STARTED, DONE, DONE_WITH_ERRORS, CANCEL, FAILED.*

1.7.2. Agent Library

The Data Transfer Agent Library is the CL implementing the API for Data Transfer. In particular the Library implements the API to contact the Data Transfer Agent Service.

Wiki page:
https://gcube.wiki.gcube-system.org/gcube/index.php/Data_Transfer_Agent

1.8. Data Transfer Portlet

The scheduler portlet is an interface implemented for the client with the general purpose of scheduling transfers. The client has also the option of monitoring and canceling a transfer. The Scheduler Portlet also provides statistics about the agent nodes so that the user can choose the most beneficial agent.

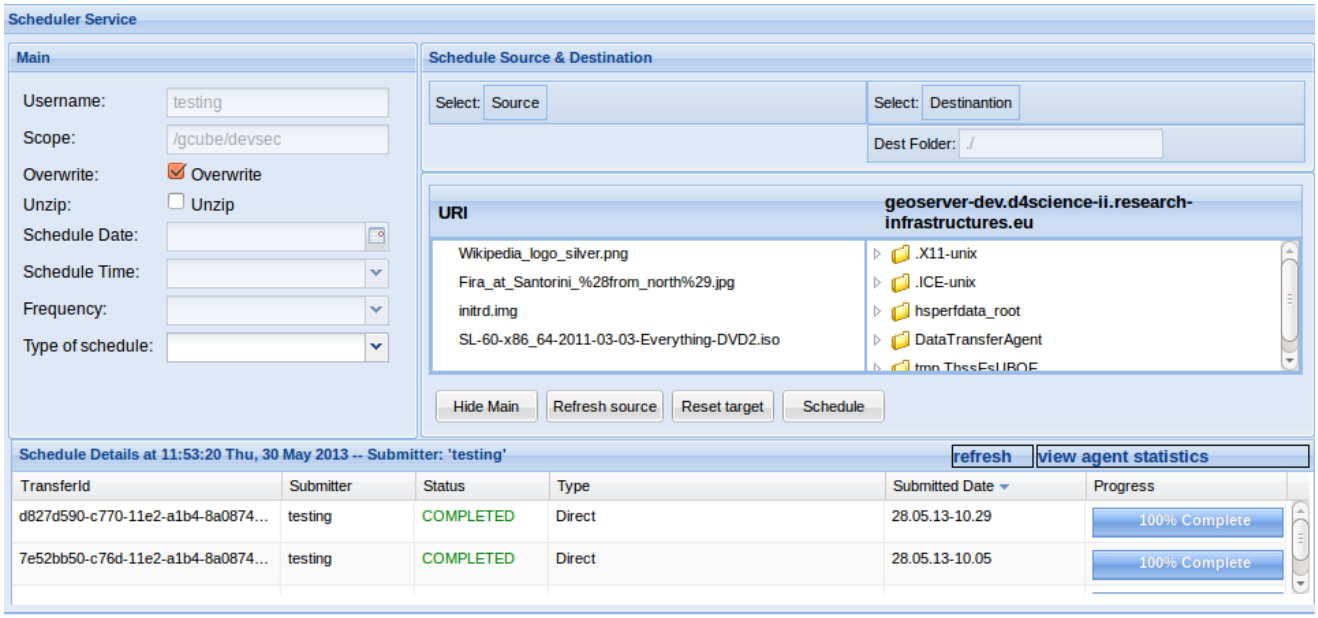


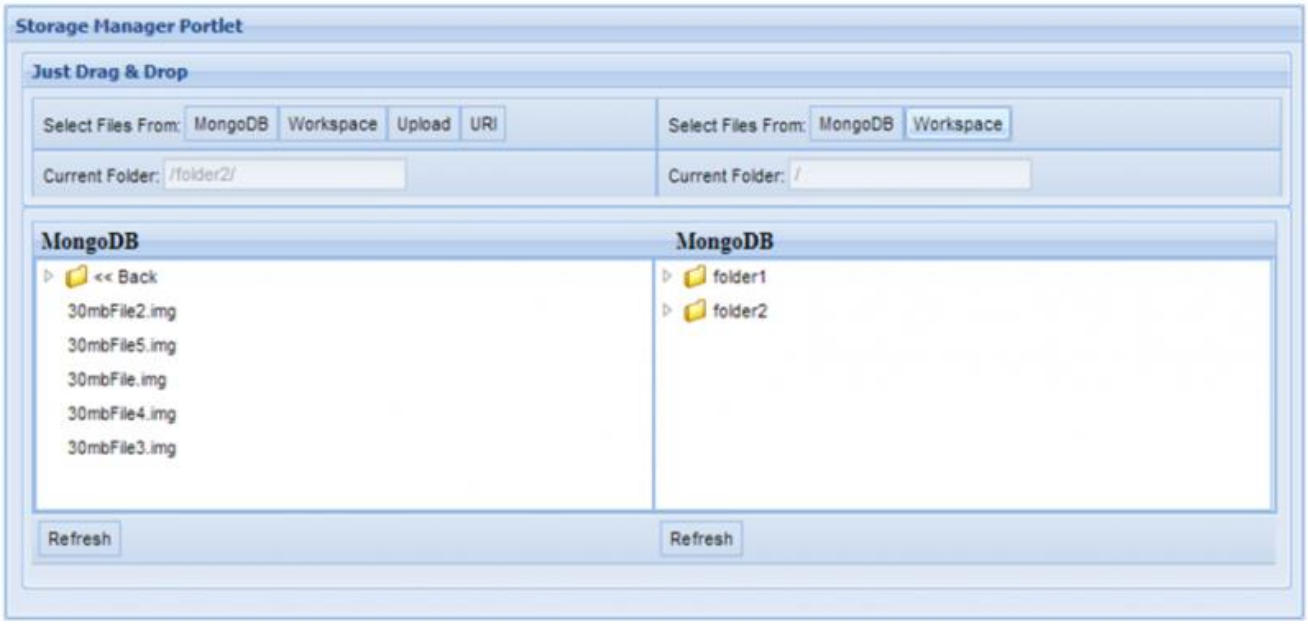
Figure 6: The Data Transfer Portlet

- Schedule, Monitor, Cancel
- View transfer outcomes
- Browsing Data Sources and Storages

Wiki page:
https://gcube.wiki.gcube-system.org/gcube/index.php/Web_Application_Scheduler_Portlet

1.9. Storage Manager Portlet

The storage manager portlet is an interface implemented for the client with the general purpose of performing transfers from/to MongoDB. The client has also the option of transferring files between Workspace and MongoDB, from URI to MongoDB and from a user uploaded file to MongoDB.



Transfer between:

- MongoDB <-> Workspace
- URI's - Uploaded files –> MongoDB/Workspace

Wiki page:
https://gcube.wiki.gcube-system.org/gcube/index.php/Storage_Manager_Portlet

2. Accounting

This section refers to the Data Infrastructure Availability, Monitoring and Accounting (T5.3) task. The main objective of this task is to define, either develop or integrate, and exploit a number of tools to efficiently monitor the status, usage, and availability of the iMarine Data e-Infrastructure. These tools provide to the appropriate infrastructure users and/or administrators the information required to control the use of the infrastructure resources and to make the infrastructure more reliable.

Part of my work regarding this task was to migrate the accounting portlet from an old third-party library of GWT to a new one and include more features.

2.1. Accounting Portlet

This portlet can be used to easily navigate accounting records by selecting one or more filters. In addition it offers the possibility to aggregate statistics, export them as CSV and create simple graphs.

The portlet has been migrated to GXT (Sencha).

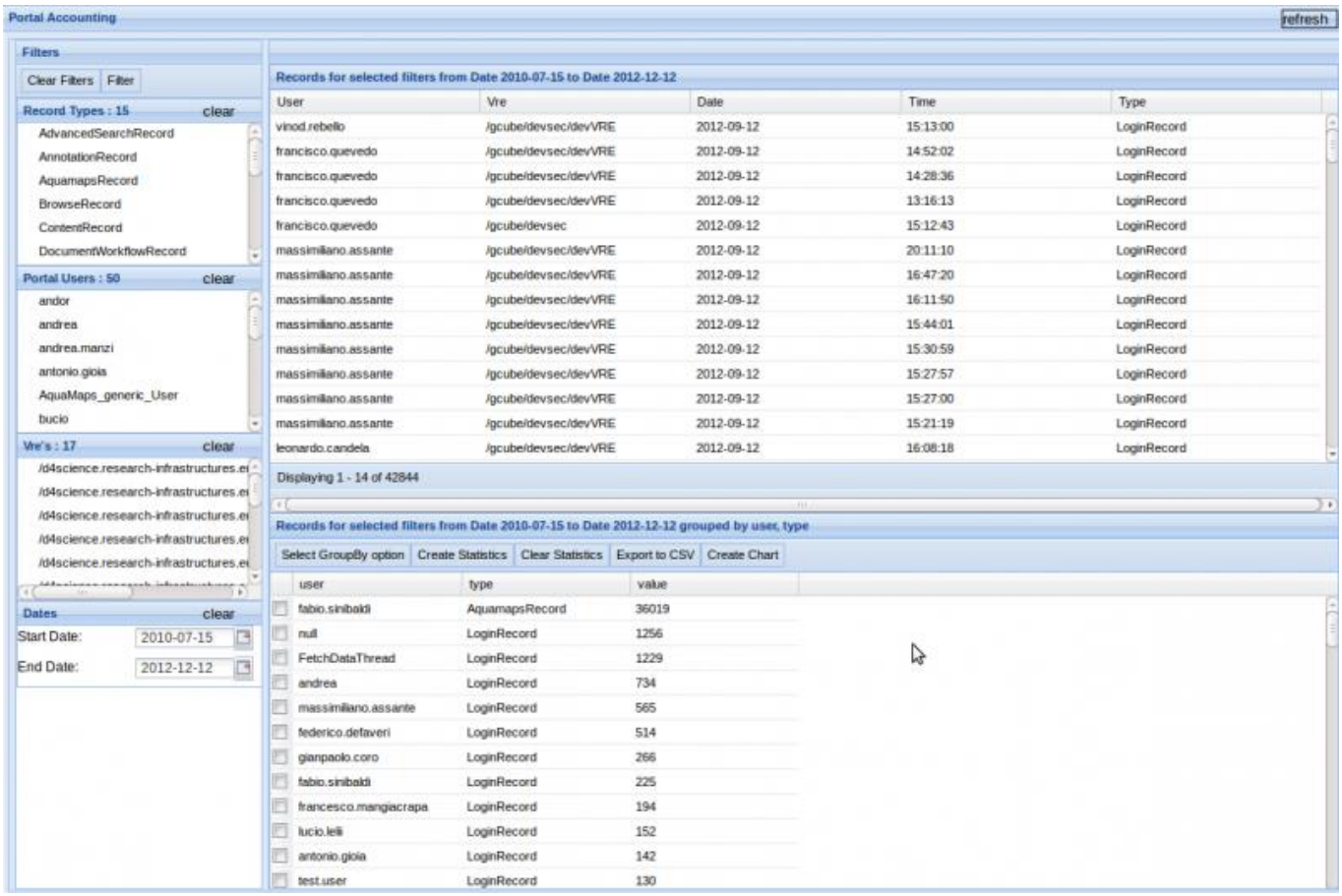


Figure 7: The Accounting Portlet

Some of the main improvements:

- Dynamically fetching data with the “live” grid panel
- Possible filtering by record type, user, scope and date
- Chart

Wiki page:
http://gcube.wiki.gcube-system.org/gcube/index.php/Messaging_Infrastructure#Portal_Accounting_portlet