J Grid Computing (2010) 8:199–221
DOI 10.1007/s10723-010-9147-y

Merging Grid Technologies

Astro-WISE and EGEE

K. G. Begeman · A. N. Belikov · D. R. Boxhoorn · F. Dijkstra · E. A. Valentijn · W.-J. Vriend · Z. Zhao

Received: 31 July 2009 / Accepted: 19 February 2010 / Published online: 10 March 2010

Abstract This paper reports the integration of the astronomical Grid solution realised in the Astro-WISE information system with the EGEE Grid and the porting of Astro-WISE applications on EGEE. We review the architecture of the Astro-WISE Grid, define the problems for the integration of the Grid infrastructures and our solution to these problems. We give examples of applications running on Astro-WISE and EGEE and review future development of the merged system.

Keywords Data Grid · Grid computing · Information system · Middleware

K. G. Begeman · A. N. Belikov (✉) · D. R. Boxhoorn · E. A. Valentijn · W.-J. Vriend
Kapteyn Astronomical Institute, University of Groningen, Landleven 12, 9747AB, Groningen, The Netherlands
e-mail: belikov@astro.rug.nl

K. G. Begeman
e-mail: kgb@astro.rug.nl

D. R. Boxhoorn
e-mail: danny@astro.rug.nl

E. A. Valentijn
e-mail: valentyn@astro.rug.nl

W.-J. Vriend
e-mail: wjvriend@astro.rug.nl

F. Dijkstra · Z. Zhao
Donald Smits Center for Information Technology, University of Groningen, Nettelbosje 1, 9747AJ, Groningen, The Netherlands

F. Dijkstra
e-mail: f.dijkstra@rug.nl

Z. Zhao
e-mail: Z.Zhao@rug.nl

1 Introduction

The growth of the data volume in many different fields of human knowledge keeps challenging our ability to provide appropriate processing of these data. The data storage and data processing problems in astronomy are comparable with the same problems in nuclear physics, meteorology, earth sciences, biology and often even exceed these disciplines in the complexity of the structure of the data which must be processed and archived.

Possibly one of the first Grid initiatives was the distributed data processing network launched by astronomical `SETI@home` project[1] which made it possible to process enormous data volumes of radioastronomical observations. Presently, there are many examples of distributed archiving and data processing projects in astronomy, including the biggest one, the Virtual Observatory.[2]

[1] http://setiathome.ssl.berkeley.edu
[2] http://www.ivoa.net

Historically, many of the Grid initiatives developed independently. They created a massive hardware and software infrastructure, and a number of projects are trying to bring together the different Grids. One example is the CROSSGRID project.[3]

The Astro-WISE information system, or Astro-WISE[4] [1], is an example of a specific Grid solution initially developed for data storage and data processing of astronomical images. Unlike CROSSGRID, the integration of Astro-WISE and EGEE is performed on the infrastructural level, and not on the level of applications only. The final goal of this integration is to make it possible for the "master" Grid infrastructure (Astro-WISE) to use resources of the other Grid (EGEE) without any changes in the already developed user applications. Such an approach saves considerable time in porting numerous astronomical applications running on Astro-WISE. In this case, Astro-WISE approaches EGEE as an external data storage and processing facilities fully integrated into the Astro-WISE system, and for EGEE, Astro-WISE is a group of applications and users.

There are many ways for the enabling Grid data storage and data processing in astronomy: development of a data processing system for ongoing and upcoming astronomical missions, for example, Planck [2] and XMM-Newton [4], creating a new hardware and software infrastructure for astronomical applications (for example, [3]), porting existing astronomical applications to the Grid. There were significant efforts in the integration between Virtual Observatory and the Grid (see, for example, [5]) which resulted in Virtual Observatory VOSpace interface.[5] Astro-WISE does not implement VOSpace but has a number of interfaces for the publishing of the data in Virtual Observatory.[6]

In this paper we consider the problem of porting Astro-WISE applications and using EGEE resources from Astro-WISE as a particular case of a much wider problem: merging different Grids. We describe the differences between Grids (Astro-WISE and EGEE) and the issues we have to address during the integration of these Grids. We review the Astro-WISE Grid structure in Section 2. In Section 3 we discuss the differences between Astro-WISE and EGEE and the reasons for the integration of both infrastructures. The solution for the merging and its practical realisation are described in Section 4. Applications which are running on Astro-WISE and that are going to profit from the integration of Astro-WISE and EGEE are reviewed in Section 5. Section 6 is devoted to the future development of an extended Astro-WISE system.

2 Astro-WISE at a Glance

The classical approach in astronomical data processing is the use of some software packages on the local PC or workstation. In this case, the researcher processes the data individually and distributes the final product only. Most of the popular astronomical software languages and packages (ITT Interactive Data Language—*IDL*,[7] European Southern Observatory Munich Image Data Analysis System—ESO *midas*,[8] Image Reduction and Analysis Facility—*iraf*[9]) assume standalone approach to work with the data. In the case of all these packages the storage of the raw and processed data is a responsibility of the user. Usually the intermediate data products are not stored. In addition, the user has to deal with different data formats for each application, converting from commonly used FITS format to a specific midas or iraf internal data format.

The storage of raw data and the final science-ready data is the duty of numerous astronomical data centers. The type of data center can vary from the archive of raw data (ESO[10]) to the "global" data centers storing the full range of science-ready catalogs for different wavelength ranges from radio to X-ray (e.g., CDS[11]).

[3]http://www.eu-crossgrid.org

[4]http://www.astro-wise.org

[5]http://www.ivoa.net/Documents/VOSpace/

[6]http://www.astro-wise.org/portal/aw_vo.shtml

[7]http://www.ittvis.com/ProductServices/IDL.aspx

[8]http://www.eso.org/sci/data-processing/software/esomidas

[9]http://iraf.noao.edu

[10]http://archive.eso.org/eso/eso_archive_main.html

[11]http://cds.u-strasbg.fr

Astro-WISE realizes a different approach by storing all data from the raw data to the final science-ready product as well as the software used for the data reduction in the same system. As a result, scientists are able to develop a new workflow or pipeline from scratch, or to use an existing recipe. They are also able to process and reprocess data without losing intermediate data products in the same system using the same environment. Astro-WISE moves from a traditional for astronomy *processing-centric* approach to a *data-centric* approach. Moving of an archive in the center of data storage and data processing is important for astronomy as this ensures preservation of the data and provides an ability to access the data from a wide range of applications.

2.1 Astro-WISE Purpose and Concept

Astro-WISE means Astronomical Wide-field Imaging System for Europe. The system was initially developed to support the data processing of the Kilo Degree Survey (KIDS[12]) on the VLT (Very Large Telescope) Survey Telescope (VST). The VST is a 2.61 m diameter imaging telescope installed on the Paranal Observatory in Chile. The instrument installed on the telescope is OmegaCAM, a large format (16k × 16k) CCD camera which will produce up to 15 TB of raw images per year. This amount is multiplied by a factor of 3 by the data processing.

Existing and upcoming telescopes will create an even larger data flow, which together with data from radio astronomy and space missions, makes the data volume to be handled by astronomy comparable with the data volume generated by nuclear physics (see, for example, LOFAR project Long Term Archive in Section 5.2).

Astro-WISE was planned as a storage and data processing system for one particular astronomical data survey, but, with time, grew up to a general astronomical information system. It is now developing into a much wider data processing information system which can be used in many other disciplines. The idea behind Astro-WISE is to keep the data model, data and data processing in the same system.

At the same time, such a system should be distributed over a number of institutes and sites, as the scientific community working with the data in the system is widely distributed and each group is coming to the system with their own resources (data storage and computing facilities). Moreover, each group is developing a part of the software implemented in the system. The users must be able to form communities (groups) to share data and data processing for the same instrument or project.

As a result, there are three principles in the data handling we implemented for Astro-WISE:

1. **Inheritance of data objects**. Using object oriented programming (Python), all Astro-WISE objects inherit key properties of the parent object, and many of these properties are persistent.
2. **Relationship between objects**. The linking (associations or references) between objects instances in the database is maintained completely. Literally it is possible to trace each bit of information back to the bits of information which were used to obtain it.
3. **Consistency**. At each processing step, processing parameters and the inputs which are used are kept within the system. The database is constantly growing by adding newly reprocessed data (an improved version of an existing object) and at the same time Astro-WISE keeps the old version and all parameters and dependences used to produce the new object from the old one.

These principles allow the reproduction of any data that has been processed in the system by any user. Combined with the requirement of distributed and multi-user data processing in the system these principles put the following requirements on the realization of Astro-WISE:

1. **a component based software engineering (CBSE)**. This is a modular approach to software development, each module can be developed independently and wrapped in the base language of the system (Python, Java) to form a pipeline or workflow.

[12]http://www.astro-wise.org/projects/KIDS/

2. **An object-oriented common data model used throughout the system**. This means that each module, application and pipeline will deal with the unified data model for the whole cycle of the data processing from the raw data to the final data product.
3. **Persistence of all the data model objects**. Each data product in the data processing chain is described as an object of a certain class and saved in the archive of the specific project along with the parameters used for the data processing.

The Astro-WISE system is realized in the Python programming language. It is allowed to wrap any program into a Python module, library or class. The use of Python also allows to combine the principles of modular programming with object-oriented programming, so that each package in the system can be built and run independently with an object-oriented data model as a glue between modules. At the same time, the logic behind pipelines and workflows in Astro-WISE allows the execution of part of the processing chain independently from the other parts. We will describe this approach in more detail in the example of optical image data processing in Section 5.1.

2.2 Astro-WISE Grid Structure, Organization and Management

Astro-WISE was designed as a system that is distributed over a number of institutes all over Europe. The design allows a number of scientists and programmers from many different institutes work together with their own storage and processing resources connected to the system. This precondition created additional challenges for the developers of Astro-WISE. This system must:

* provide a common environment for a number of scientists for the development of data reduction programs,
* distribute data and metadata over a number of geographically remote sites,
* allow for adding new Astro-WISE sites to the system without interrupting the system,
* allow a number of users to work on the same data chunk simultaneously,
* allow to share data among a number of users and restrict this chunk of data from public access,
* bring together various data storage and computational facilities provided by institutes participating in the system.

Thus, this specification defines a Grid system from the point of view of joining physical resources that are not under central administrative control. The system consists of the following elements:

* Data storage nodes (dataservers) are used to store all data files created by the users (and by the programs run by them).
* A metadata database to store a full description of each data file with links and references to other objects.
* Astro-WISE programming environment to provide a command line interface to the system and a number of web interfaces which give the user an ability to have an easy and fast access to the stored data and data processing.
* Computing nodes for the data processing.
* A version control system for developers to include new modules, classes and libraries into the system.

The list of elements above represents a complete deployment of an Astro-WISE site (see Fig. 1), while the actual configuration of a site can include only a subset of these elements. This is up to the participant of the system to decide the exact configuration of the site. Presently Astro-WISE includes sites at Groningen University, Leiden University and Nijmegen University (The Netherlands), Argelander-Institut für Astronomie, Bonn and Universitäts-Sternwarte München (Germany) and Osservatorio Astronomico di Capodimonte, Napoli (Italy).

The principle of Astro-WISE is to store bulk data (images, for example) in separate files, each file belongs to a data object in the database. The full and complete description of the object is in the metadata database along with the path to the corresponding file. We define the metadata as the description of the information stored in the file. The actual level of this description depends on the data model implemented in each particular case,

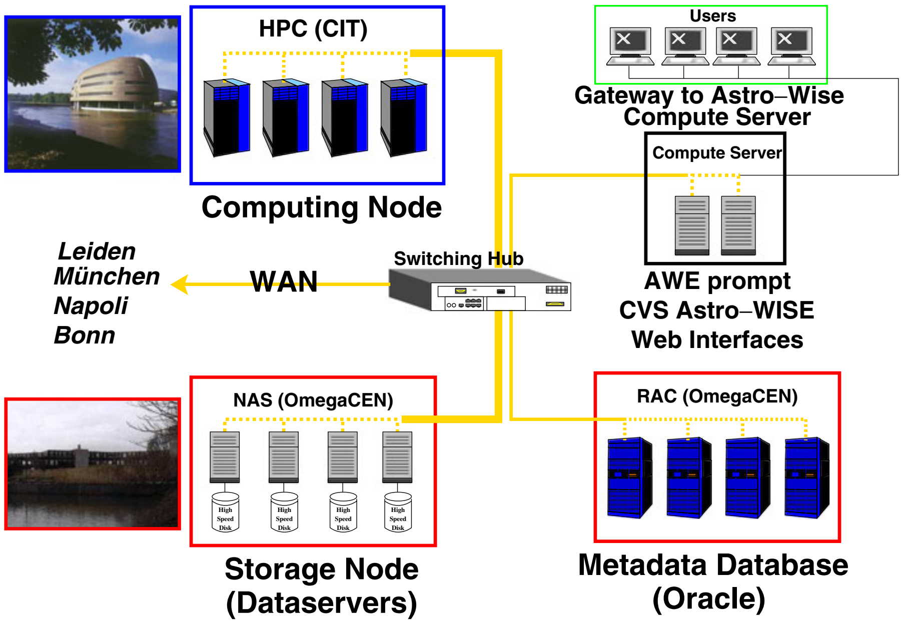


Fig. 1 Typical architecture of Astro-WISE site in full mode (data storage node, computing node, web interfaces, CLI and version control system)

for example, for the optical images processing pipeline (see Section 5.1), the metadata includes all parameters related to the image except the image itself (i.e., pixel data). As soon as a user performs processing on this data object a new object will be created and stored with a link to the parent object.

Following the definition of Grid given by Ian Foster[13] [21], Astro-WISE *coordinates resources that are not subject to centralized control* (each Astro-WISE node is an independent one managed by the local authorities only), *uses standard, open, general-purpose protocols and interfaces* (Astro-WISE built upon http for data transfer and https for authorization for web interfaces), *delivers non-trivial quality of service* (see Section 5 for the detailed description). We would like to underline, that the original Astro-WISE does not use any standard Grid protocols and interfaces. Merging of Astro-WISE and EGEE allowed us to use some of these protocols and interfaces from Astro-WISE (see Section 4).

We will continue with the description of each component of Astro-WISE in the following sections.

[13] http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf

2.3 Data Server

The Astro-WISE approach to scalable storage is to use dataservers to store large files with n-dimensional data and databases to keep track of all metadata. Practically any type of data can be stored in these files, from raw images to text files. Each file on a dataserver has a unique name, this unique name is stored in the database as part of the metadata.

The dataservers are grouped per geographical location. Within each group, all dataservers know about the existence of all other dataservers in the group. Likewise, a dataserver is aware of the other groups. Clients access a dataserver using the standard HTTP protocol to retrieve a file using a URL. The dataserver can then either return the requested file or forward the client to the dataserver that has the file. The dataservers communicate with each other using the HTTP protocol. The dataservers support storing and retrieving files, authenticated deletion of files and caching of files. In addition, a file can be compressed or decompressed on-the-fly during retrieval, and a slice of the data can be retrieved by specifying the slice as part of the URL.

The underlying hardware for the dataservers can use any operating system and filesystem that support long case-sensitive filenames and that is

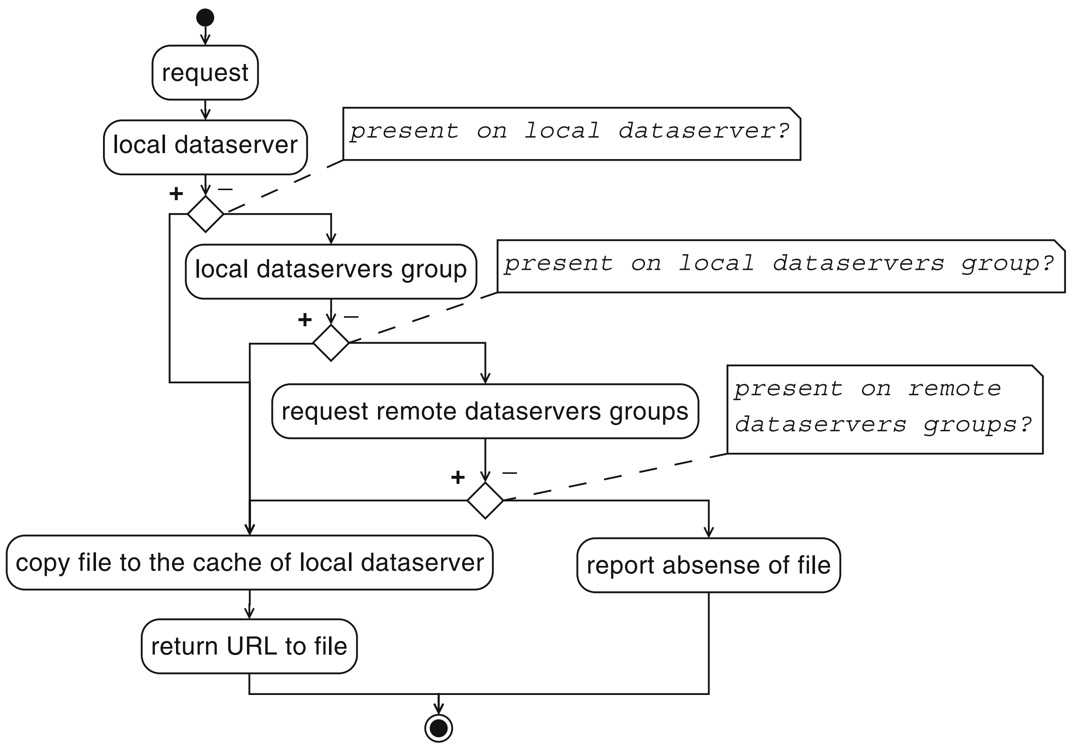

request
local dataserver
present on local dataserver?
+
–
local dataservers group
present on local dataservers group?
+
–
request remote dataservers groups
present on remote dataservers groups?
+
–
copy file to the cache of local dataserver
report absense of file
return URL to file

Fig. 2 The logic of the search for the file on dataservers

scalable to very large files and directories. The only communication with external clients takes place via the HTTP protocols that the dataserver provides. For the Astro-WISE dataservers Linux and XFS are in use as operating system and filesystem.

The user of the system does not know the actual location of the file on the underlying file system and operates with the URL of the file only. The URL has the form http://<data server address>/<file name>. Each Astro-WISE node has a predefined "local" dataserver which the administrator of the node believes to be most optimal for the data retrieval for this node. In fact there is a "local" group of dataservers managed by the same authority, all these dataservers are connected with each other. By user request the unique file name is checked in the cache space of the dataserver of the "local" group. In the case the file is not found, the dataserver checks its own storage space, and if there is no such file located, it requests the file from all "local" data servers. In the absence of the file, other groups of dataservers are requested, and as soon as the file is found, it is copied from the remote dataserver to the cache of the "local" one (see Fig. 2 for the algorithm of the search, and Figs. 3 and 4 for the actions

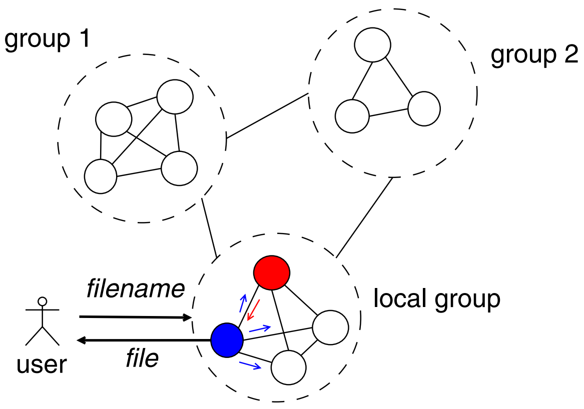


Fig. 3 Search for the file on the local dataservers group, *blue arrows* are requests for the file, *red arrow* is a copy operation of the requested file from the remote dataserver to the cache of the local one

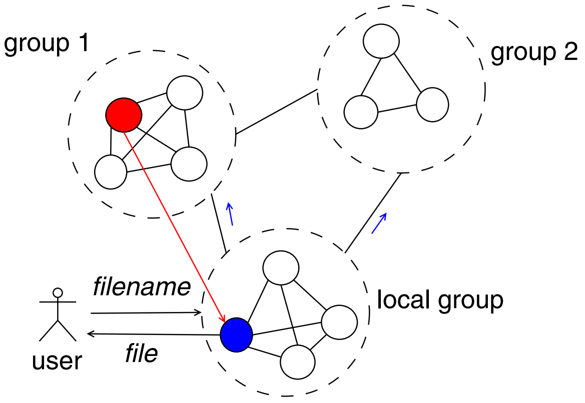
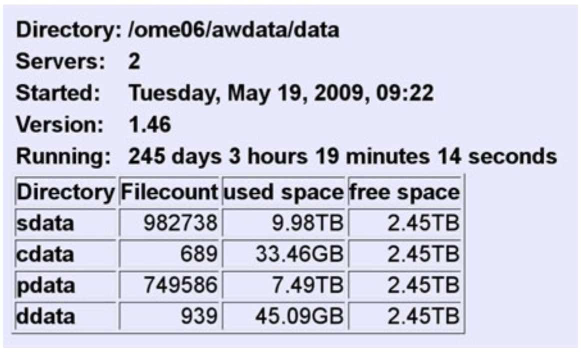


Fig. 4 Search for the file on remote dataservers groups

the local dataserver performs to find a requested file). We use "local" in quotes to emphasise that the "local" dataserver may be remote from the user, actually, there can be no local dataservers installed next to the user (for example, the user in Groningen can select the dataserver in Munich as a "local" one). Figure 5 shows the status of one of the "local" group of dataservers (Groningen) and Fig. 6 represents the load of one of the dataservers in the "local" group.

2.4 Metadata Database

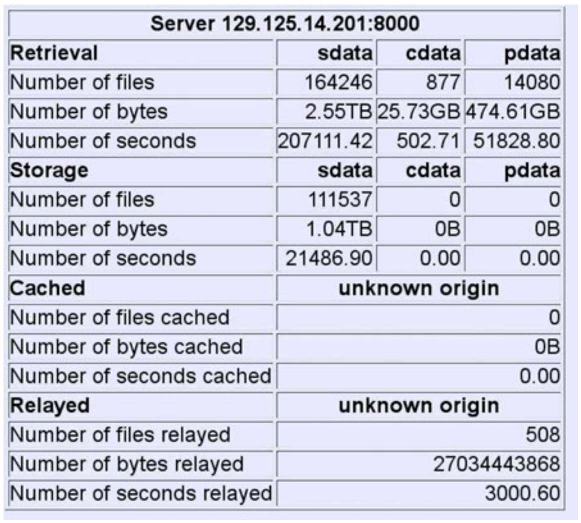
The data items stored in Astro-WISE as a file are registered in the Astro-WISE metadata database with the unique filename which can be used to retrieve the data item from the system. Along with



Directory: /ome06/awdata/data
Servers: 2
Started: Tuesday, May 19, 2009, 09:22
Version: 1.46
Running: 245 days 3 hours 19 minutes 14 seconds

Directory	Filecount	used space	free space
sdata	982738	9.98TB	2.45TB
cdata	689	33.46GB	2.45TB
pdata	749586	7.49TB	2.45TB
ddata	939	45.09GB	2.45TB

Fig. 5 An example of the local group dataserver load (dataserver status page is shown)



Server 129.125.14.201:8000			
Retrieval	**sdata**	**cdata**	**pdata**
Number of files	164246	877	14080
Number of bytes	2.55TB	25.73GB	474.61GB
Number of seconds	207111.42	502.71	51828.80
Storage	**sdata**	**cdata**	**pdata**
Number of files	111537	0	0
Number of bytes	1.04TB	0B	0B
Number of seconds	21486.90	0.00	0.00
Cached	**unknown origin**		
Number of files cached	0		
Number of bytes cached	0B		
Number of seconds cached	0.00		
Relayed	**unknown origin**		
Number of files relayed	508		
Number of bytes relayed	27034443868		
Number of seconds relayed	3000.60		

Fig. 6 An example of the load of the dataserver in the local group

the name of the file, the data model for the item is stored in the metadata database. This includes all persistent attributes specified by the model. The attributes with the parameters used for the data processing are also part of the model.

The Persistent Object Hierarchy forms the core of the data lineage and processing in the Astro-WISE environment. Processing is done through invocation of methods on these objects before they are made persistent. Since the objects are made persistent recursively, all operations and attributes of the objects are implicitly saved into a database. As a result the metadata database allows an intensive search on attributes of an object or its dependencies. This is used to avoid partial reprocessing of data if the object with desired processing parameters already exists and the user has access rights to this object. The detailed description of this feature of Astro-WISE is done in [23].

All the I/O of processes makes use of the database. This includes all information about the processing of data and the processing history. The database stores all persistent objects, attributes and raw data either as fully integrated objects or as descriptors. Only pixel values are stored outside the database in images and other data files. However, their unique filenames are stored in the database. Data can therefore only be manipulated through interaction with the database. A query to the database will provide all information related to the processing history and to the unique names of all stored associated files, attributes and objects. Thus, the system provides the user with transparent access to all stages of the data processing and thereby allows the data to be re-processed and the results fed back into the system. We utilize inbuilt security and authorization features of the database to enable sharing data in the system without breaking the dependency chain. The authorization is implemented on the level of the metadata database, that is, any user logging in the system with any interface (web-service, CLI) is checked in the database for his password, privileges, user group and access rights.

Access to the database is provided through Python. Python is used for both the Data Definition Language and the Data Manipulation Language. The database interface maps class definitions that are marked as persistent in Python to the corresponding SQL data definition statements.

2.5 Computing Resources

The Astro-WISE system contains a sophisticated distributed processing system. It is a fully scalable system that will overcome the huge information avalanche in, for example, wide-field astronomical imaging surveys. This system allows the end-user to process raw science and calibration data. It also maintains a history of dependencies that can be followed from the end data product back to the raw observational data.

The distributed processing unit (DPU) sends jobs to parallel clusters or single machines. It can automatically split up a processing job in parallel parts that run on different nodes. It allows users to run their own Python code. The system runs on *openpbs* or under its own queue management software. A DPU allows for synchronizations of jobs and can also transfer parts of a sequence of jobs to other DPU's.

The user can select a cluster or a single node which will be used for the data processing, and the administrator of the Astro-WISE nodes sets a processing node as default.

2.6 Applications, Interfaces and Programming Environment

The main language for the system is Python, but each user can develop his own application or use an existing application which should be wrapped into Python. Usually, users develop pipelines or workflows using existing "blocks" with the help of pre-defined Python libraries and classes. Users can also change an existing data model if necessary or implement a new one.

The Command Line Interface of Astro-WISE, the AWE (Astro-WISE Environment), can be installed on a site without any other components of Astro-WISE (data server and metadata database). Basically the AWE prompt is a link to a local Python installation with the Astro-WISE libraries and environments.

Apart from the AWE prompt Astro-WISE supports a range of web interfaces. This makes it possible for a user to work with data stored in Astro-WISE without the AWE prompt, but with the use of the world wide web only. The web interfaces are divided into two types: data browsing/exploration and data processing/qualification. The first group includes:

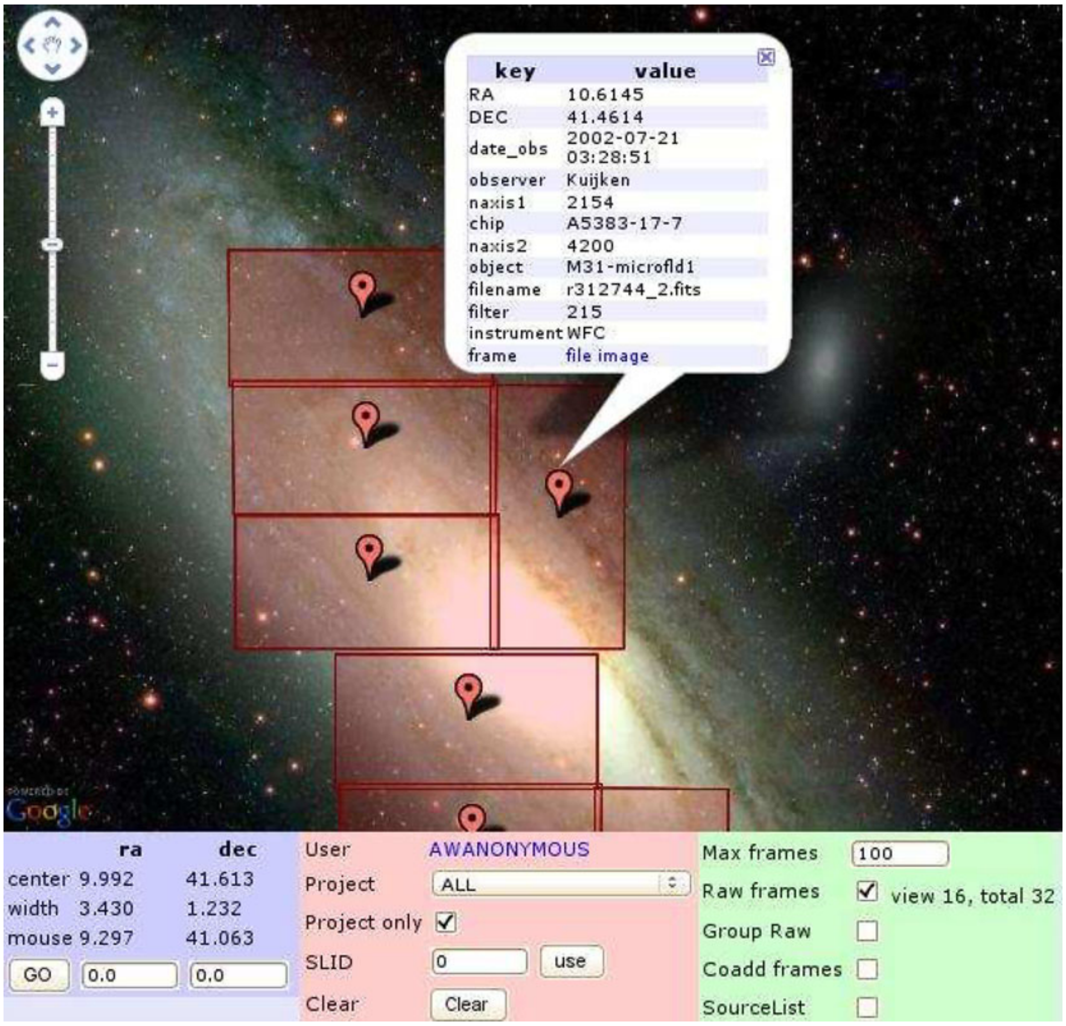
- dbviewer[14]—the metadata database interface which allows to browse and query all attributes of all persistent data classes stored in the system,
- quick data search[15]—allows to query on a limited subset of attributes of the data model (coordinate range and object name), and provides results of all projects in the database,
- image cut out service[16] and color image maker[17]—these two services are for the astronomical image data type and allows to create a cut out of the image or to create a pseudo-color RGB image from three different images of the same part of the sky,

[14]http://dbview.astro-wise.org

[15]http://gowise.astro-wise.org

[16]http://cutout.astro-wise.org

[17]http://rgb.astro-wise.org



key	value
RA	10.6145
DEC	41.4614
date_obs	2002-07-21 03:28:51
observer	Kuijken
naxis1	2154
chip	A5383-17-7
naxis2	4200
object	M31-microfld1
filename	r312744_2.fits
filter	215
instrument	WFC
frame	file image

	ra	dec
center	9.992	41.613
width	3.430	1.232
mouse	9.297	41.063
GO	0.0	0.0

User AWANONYMOUS
Project ALL
Project only ✓
SLID 0 use
Clear Clear

Max frames 100
Raw frames ✓ view 16, total 32
Group Raw ☐
Coadd frames ☐
SourceList ☐

Fig. 7 The GMap interface with M31 image from GoogleSky and images stored and processed in Astro-WISE available for this part of sky

* GMap[18]—exploration tool of the Astro-WISE system using the GoogleSky interface (Beta, see Fig. 7).

Data processing/qualification interfaces are:

* target processing[19]—the main web tool to process the data in Astro-WISE. This web interface allows users to go through pre-defined processing chains, submitting jobs on the Astro-WISE computing resources with the ability to select the computing node of Astro-WISE,
* quality service[20]—allows to estimate the quality of the data processing and set a flag highlighting the quality of the data,
* CalTS[21]—web interface for qualifying calibration data.

[18]http://skymap.test.astro-wise.org
[19]http://process.astro-wise.org
[20]http://quality.astro-wise.org
[21]http://calts.astro-wise.org

The exact set of web interfaces depends on the data model implemented in the system. The web interfaces described above are for the optical image data processing and follow the requirements for this particular data model and data processing chain. Astro-WISE allows the implementation of new web interfaces for the data model and the data processing defined by the users. The developer of the new web interface will use predefined classes and libraries of Astro-WISE to create it.

3 Merging EGEE and Astro-WISE: Why Do We Need It?

The main reason for merging Astro-WISE and EGEE is to make Astro-WISE available to a larger user community, especially users who are already using EGEE resources for data storage and processing. The LOFAR Long Term Archive (LTA, see Section 5.2) will, for example, be built on top of the EGEE Grid as well as on Astro-WISE infrastructure. Astro-WISE will be used for storing metadata, processing parameters and for the organisation of a number of processing pipelines. EGEE will be used to store a part of the LTA data and to process it. To make this possible, we are going to extend the abilities of Astro-WISE by including EGEE storage in the Astro-WISE system and allowing processing of the data from Astro-WISE and with Astro-WISE pipelines on EGEE. At the same time, we will keep the way Astro-WISE works with metadata and the organisation of the processing intact, because these are the key advantages of the system. As result, the storage is distributed over Astro-WISE storage nodes (dataservers) and EGEE storage nodes. The second reason for merging Astro-WISE and EGEE is the introduction of international standards used by the EGEE Grid into the Astro-WISE system which makes any future connection to other Grids easier.

The comparison between the EGEE and Astro-WISE approaches in Section 3.2 will point out the main problems encountered in the merging of the two Grid solutions.

3.1 EGEE

The well known EGEE project[22] [6] is currently operating a worldwide Grid infrastructure consisting of more than 250 sites in more than 50 countries. The main goal of the project is to provide researchers with access to a production level Grid infrastructure. The main focus points are a continuous operation of the infrastructure 24 h per day, support for more user communities and addition of more resources. The size of the infrastructure as well as its use have been increasing over the last years.

Note that in this paper we will refer with the name EGEE more to the international infrastructure operated by the project than to the project itself, which will end in 2010. We expect that the infrastructure itself will continue to be relevant after the project has finished.

The EGEE Grid is built on the gLite middleware stack[23] [7]. This middleware combines components from a number of sources, including LCG[24] (Large Hadron Collider Computing Grid), VDT[25] (Virtual Data Toolkit), which is also used by Open Science Grid[26] [8] and contains software from Globus[27] [9] and Condor[28] [10], and from gLite itself. The middleware tries to make use of well defined international standards, where it is possible. A main source for these standards is the Open Grid Forum community.[29]

3.2 EGEE and Astro-WISE: Comparison

Within the context of this paper, it is relevant to compare the nature of Astro-WISE software and

[22]http://www.eu-egee.org

[23]http://www.glite.org

[24]http://lcg.web.cern.ch/LCG/public

[25]http://vdt.cs.wisc.edu

[26]http://www.opensciencegrid.org

[27]http://www.globus.org

[28]http://www.cs.wisc.edu/condor

[29]http://www.ogf.org

infrastructure with that of the EGEE Grid middleware and infrastructure. For this comparison, there are a few points to focus on, such as the use of standards and security, the target audience and the main goals.

One of the differences that has already been hinted at in the previous section describing EGEE, is that the gLite middleware tries to built on international Grid standards, if it is possible. Astro-WISE, on the other hand, has been developed to serve a specific purpose, and has developed its own techniques where needed based on the standard protocols and interfaces (non-Grid protocols and interfaces). It uses an object-oriented approach and Python language for the general framework, and the standard HTTP protocol to access the dataservers.

Another difference is the way authorisation is implemented. In Astro-WISE the database plays a key role. The access rules for the data are stored in the database, as well as the user accounts and project information. This makes managing the privileges on the data much easier than in the case of the EGEE Grid where the authorization is distributed over all the services, i.e., in the case of EGEE changing access rules to the data is rather difficult because these have to be enforced on all the storage servers. Another point worth noting with respect to security is that the EGEE Grid security is based on X.509 certificates whereas Astro-WISE users need just a username and password for accessing the services.

The final and probably most important difference is that EGEE focuses on the basic infrastructure components providing computation and storage and the operational aspects of those. The focus of Astro-WISE is on the construction of a complete system integrating the metadata database with the processing and storage facilities. Astro-WISE provides a layer on top of the Grid infrastructure that keeps track of the data as well as the processing that is performed on it. This is then all presented to the user through a uniform interface that hides the underlying complexities. This is something that gLite does not provide. The user will need external services on top of the EGEE Grid to perform tasks like this.

Although services for managing analysis jobs (GANGA[30] [12], PanDa[31] [11], etc.) as well as file catalogs (LCG File Catalog[32]) and metadata databases (AMGA[33] [13]) have been developed around the EGEE infrastructure, none of them allows the full tracking of the processing and data streams that Astro-WISE is capable of. Moreover, AMGA allows a hierarchical representation of the metadata grouped in a tree-like structure following the "directory-file" approach, while Astro-WISE allows an implementation of a complex object-oriented metadata model with links between attributes and entities.

3.3 EGEE and Astro-WISE: Concept of Integration

The requirement which comes from the user of the system (e.g., LOFAR LTA) is to combine the Astro-WISE ability to handle metadata and store processing parameters with the data storage and processing facilities of EGEE. It is not easy to meet this requirement because of different approaches used in the authentication and authorisation by Astro-WISE and EGEE as well as different protocols for data access and transfer.

To keep the functionality of Astro-WISE, it is necessary to keep the link between metadata database and implementation of data model with persistent attributes in Python libraries and classes. Python serves as a wrapping language for any application running in the system.

As EGEE has a lot to add to storage and computing resources and nothing to add to Astro-WISE metadata handling EGEE will be included into Astro-WISE as a collection of new "Astro-WISE" storage and computing nodes. To access EGEE resources, an Astro-WISE user will need

[30]http://ganga.web.cern.ch/ganga

[31]https://twiki.cern.ch/twiki/bin/view/Atlas/PanDA

[32]https://twiki.cern.ch/twiki/bin/view/LCG/LfcGeneralDescription

[33]http://amga.web.cern.ch

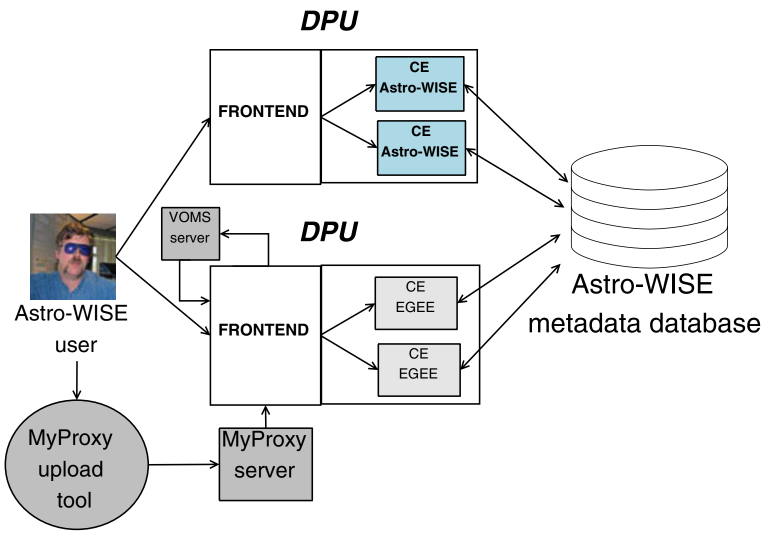


Fig. 8 The use of the proxy server for job submitting on EGEE

to obtain a Grid certificate (i.e., to become an EGEE user, or, as an option, to use an Astro-WISE service which relies on a robot certificate). In the case of such service, the user identifies himself via Astro-WISE system and Astro-WISE service uses a robot certificate to access Grid resources.

4 EGEE and Astro-WISE Integration: Solution

There are three main areas in the integration of Astro-WISE and the EGEE Grid. The first area is that of the authentication and authorisation of users. Astro-WISE makes use of accounts protected by a username and password. The Grid makes use of X.509 certificates for user authentication. The handling of X.509 certificates needs to be integrated in the Astro-WISE environment.

The second area is job submission. Astro-WISE submits jobs to it's own compute cluster via Astro-WISE distributed processing unit (DPU), which has been adapted to be able to submit jobs to the Grid as well. This job submission via DPU has been integrated with the authentication using X.509 certificates.

The last area is the use of Grid storage. The Astro-WISE storage system has been extended to be able to handle Grid storage using the SRM protocols.

4.1 Authentication and Authorization

The authentication of Astro-WISE using username and password needed to be integrated with the X.509 certificates used on the Grid. In general the use of X.509 certificates can be a stumble block for users. Furthermore, there is an existing Astro-WISE user community that is used to working with a username and password only. Therefore a solution was needed that integrates nicely with the current methods.

For obtaining a proxy certificate from the user on the Astro-WISE DPU server two methods can be used. The first one is to make use of proxy delegation, the second is to use a MyProxy server as an intermediate machine. We have opted to make use of a MyProxy server because that solution is easier to implement and also a recommended way of working with a Grid portal.[34] See Section 4.3 and Fig. 8 for further details.

The second step was to choose a lightweight client that can be used for storing a proxy in the MyProxy server that does not depend on a large software stack to be installed. The latter would be contrary to the idea of having a Python environment for Astro-WISE that will run almost

[34]http://grid.ncsa.illinois.edu/myproxy/portals.html

anywhere. As a temporary solution, we deployed Astro-WISE on a gLite UI installation, but that was found to be too cumbersome to install at other locations, and managing login accounts for all users centrally is also not an option.

Because Astro-WISE is a Python environment, adding a Python implementation of the MyProxy client tools would be the best solution. Given the limited manpower developing a solution from scratch was not considered. When we started looking, we were not able to find a complete enough and easy to use implementation. The most promising solution was a NDG Security development[35] [15] within the OMII-UK[36] and the NERC Data Grid[37] [16] projects. When we reconsidered this solution recently it was found to be much more mature and we now plan to integrate their MyProxyClient[38] into Astro-WISE.

A second option is to let users upload a Grid proxy to a proxy server through their browser. The UK National Grid Service has developed a Java Web Start tool to do this.[39] We have taken this tool and adapted it to the local environment by adding the national CA certificate and a local proxy server. Another change is that the tool has been adapted to also upload credentials using the DN of the user to allow proxy renewal for long term jobs.

In order for the Astro-WISE services to retrieve a Grid credential, the user needs to store the proxy on the MyProxy server using the same username and password as used for the Astro-WISE system.

4.2 Connection to EGEE Computing Facilities

To enable the DPU access to EGEE computing, some major changes to the server frontend had to be implemented. The main changes were in how to submit the job, how to monitor the job and how to do the Grid-authentication in a user-friendly way. Another problem is that the Astro-WISE software must be installed on the assigned EGEE compute nodes. For this, pre-compiled, self-extracting packages were made of the Astro-WISE software, which will be downloaded to the EGEE-nodes and executed before running the actual Astro-WISE job. For EGEE resource centers which allow us to install the Astro-WISE software locally, this step is of course not needed.

Because Astro-WISE is able to use multiple CPUs to exploit inherent parallelism, parallel jobs can be submitted to the EGEE Grid. The use of the parallel job type has the benefit of automatic synchronisation of the start time of the processes. It also causes the job to run at a single site using similar CPU cores. This makes the job run more efficiently when a communication step is needed in between. All communication between the parts of the job is done through the database, making it independent from parallelisation support libraries like MPI or *pthreads*.

The authentication problem was solved by using a MyProxy Upload Tool[40] which should be run by the user once every two weeks. These proxies are authorized by the Astro-WISE username and password. Whenever a user submits a job to the EGEE DPU, the DPU retrieves the user credentials secured with the Astro-WISE username and password from the MyProxy server.

After the DPU has obtained a proxy from the myproxy server the required VOMS [14] credentials are added to the proxy on the fly.

4.3 Connection to EGEE Data Storage Facilities

The Astro-WISE storage system has been changed to handle data stored on the Grid using SRM[41] (Storage Resource Manager). SRM is a protocol for Grid access to mass storage systems. The protocol itself does not do any data transfer, instead it asks a Mass Storage System (MSS) to

[35]http://www.omii.ac.uk/wiki/NDGSecurity

[36]http://www.omii.ac.uk

[37]http://ndg.nerc.ac.uk

[38]http://proj.badc.rl.ac.uk/ndg/browser/TI12-security/trunk/python/MyProxyClient

[39]http://www.ngs.ac.uk/tools/certwizard

[40]http://myproxy.egee.astro-wise.org

[41]https://sdm.lbl.gov/srm-wg/doc/srm.methods.v2.0.doc

make a file ready for transfer, or to create space in a disk cache to which a file can be uploaded. The file is then transferred using a Transfer URL or TURL. In the paper we use *srm* to designate the file identifier in the SRM system, and *SRM* is used for the definition of the protocol.

As explained in Section 2.3, Astro-WISE uses dataservers to store large files. Each file on a dataserver has a unique name, and this unique name is stored in the database as part of the metadata. In Astro-WISE, the Python class which represents objects that have an associated data file is called a DataObject, that is, every instance of class "DataObject" or every class which is derived from class "DataObject" has an associated data file.

Handling data stored on the Grid in Astro-WISE requires knowledge about the protocol and the location of the data. Both the protocol and the location can be defined in a URI, Uniform Resource Identifier. An URI is of the form `protocol://server/path/file`, where `protocol` can be *http*, *srm* or *gpfs*. Given an URI, the "DataObject" class can take the appropriate actions for storing and retrieving the file, given the protocol. The URI must be stored in the database.

In order to connect to the EGEE data storage, additional interface and logic need to be added to the existing system to deal with multi-protocol storage access.

First of all, an interface to the EGEE Grid is needed to retrieve and store data from/to the Grid. The interface needs to be able to figure out whether a user has a valid proxy, knows how to generate and renew a proxy given a certificate, and be able to store and retrieve files to or from the Grid using certain protocols.

Second, if multiple protocols can be used to retrieve or store a file, the system needs to be able to provide an interface to allow users to select a protocol to use, or be able to retrieve or store a file using its default logic. For example, a user may prefer to retrieve or store a file only from or to the Grid, or he/she doesn't care where the file is retrieved or stored at all.

Finally, due to security reasons, the system needs to provide a mechanism which separate the access right of the DataObject and its associated data files. This is required by applications such as LOFAR, where different groups or users may have different privileges to retrieve the associated files.

A general solution to address the issue of access right is to use a separate (Storage) table to store the URI of a file. The URI stored in the Storage table should be constant in time, such that we don't have to update this when, for example, the file is moved from disk to tape. The advantages of this solution are as follows:

- The Storage table can have multiple entries for the same object, therefore we can administrate multiple copies (backups) of files.
- The privileges of the Storage table can be different from the privileges of the rest of the metadata, which adds an extra security layer to the system. For example, everyone can see the metadata of files, but only privileged users can see the URI.
- The framework can easily be extended to support other storage systems by adding the corresponding interface.

In the EGEE Grid, the LFC is a central component. It keeps track of the location of files over the Grid. All files are uniquely identified with a Grid Unique IDentificator (GUID). Multiple, human readable, logical file names (LFN) may point to this GUID. Each file may be stored at a number of physical locations. In principle Astro-WISE could fulfill the same role using its internal databases. A choice therefore has to be made to use the LCG file catalog or not.

In order to retrieve files from the Grid, either the GUID or an *srm* of a file should be provided. Therefore, one of them should be kept in the Storage table. The GUID is the unique identifier of a file, while an *srm* is the storage URL of a copy of the file. If the GUID of a file is stored in the Storage table, the LFC is needed to look up the multiple copies of the file.

The main disadvantage of using the LFC is that it is another component that has to be operated and maintained, thus becoming crucial for the operation of some Astro-WISE applications such as LOFAR. The advantage is that a set of high

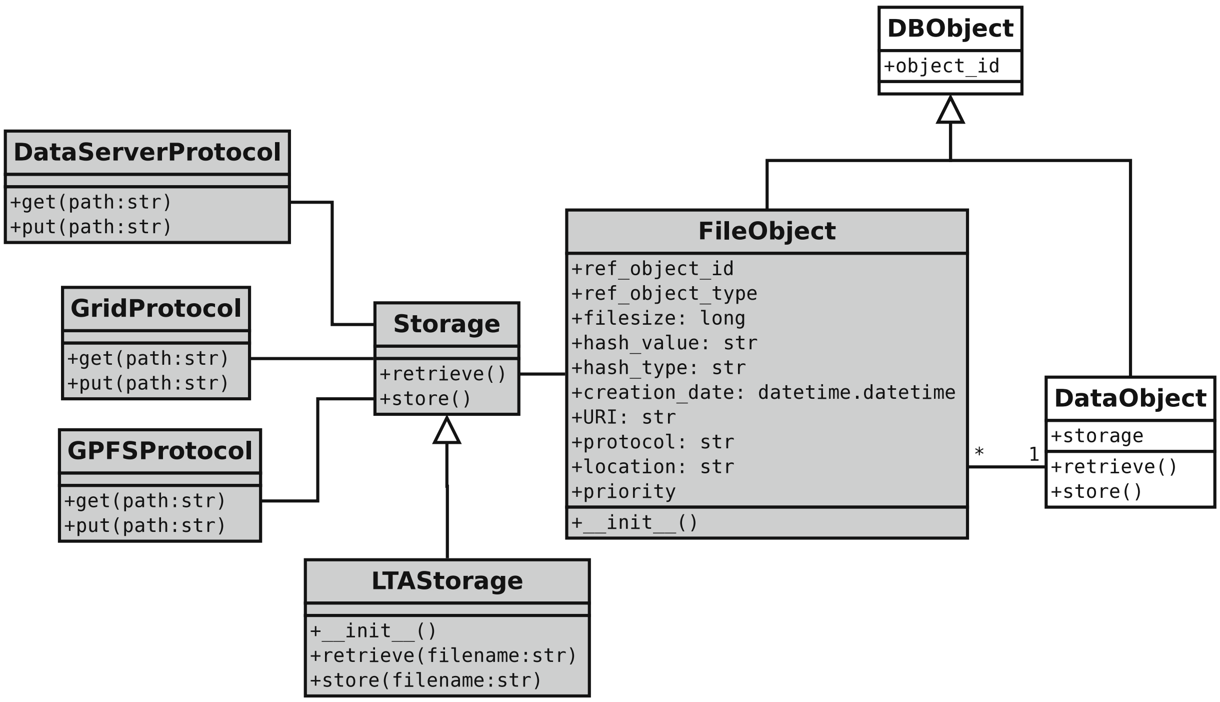


Fig. 9 Class diagram of the newly designed classes of Astro-WISE to support storage of data files on EGEE and Global File Systems

level tools exist for using the LFC. These tools would otherwise have to be replicated within the Astro-WISE environment.

The advantages of storing srms in the Storage table are as follows:

– Multiple *srms* of a file can be stored in the table, which makes it easier to look up how many copies of the file exist.
– SRM commands can be used to store and retrieve data from the Grid directly, using these, we avoid the dependencies on the LCG File Catalogue, therefore making the system more robust.

Based on the above analysis, we decided to use a separate table to store the srms of files that are stored on the Grid. The detailed design of the framework and how it is connected to the rest of the system are shown in Fig. 9.

In Astro-WISE, a DataObject is an object that has an associated data file, which is identified by the unique object identifier (`object_id`). The Storage table we use to store the associated data files is called the "FileObject" class.[42] The object identifier and the object type of a DataObject are used as reference to identify the relationship between the data file and the DataObject. In the Storage table, a FileObject contains information of a file, such as filesize, creation date, hash value, and URI, etc. The abstract Storage class in the framework defines a general interface to the different protocols. It defines a retrieve and a store method, which should be implemented by the different kinds of storage, that is, classes that inherit the Storage class. For example, in Fig. 9, class LTAStorage (the LOFAR Long Term Archive Storage) inherits from the Storage class. It instantiates the available protocol interfaces (e.g., the file protocol and the Grid protocol), and implements the retrieve and store methods with a

[42]FileObject is the current name of the class and may change in the future.

default logic. The default logic for retrieving a file used in the class "LTAStorage" is as follows:

1. Search in the local dataserver to see whether a file is locally available.
2. If so, the DataServerProtocol interface is used to retrieve the file.
3. Otherwise, search in the Storage table, and retrieve the nearest available file using the corresponding protocol.

The different protocol classes implement the concrete action of getting and putting data from or to the storage (see Fig. 10). They perform the actual retrieve and store tasks according to the type of the storage.

Figure 10 shows how a LOFAR user retrieves a file from the EGEE Data Storage via Astro-WISE. For example, the user wants to retrieve a file associated with DataObject `data_obj`. He can use the retrieve method from the DataObject class to retrieve the file, i.e. `data_obj.retrieve()`. The DataObject 'retrieve' method in turn calls the corresponding retrieve method from a Storage object, in this case, the LTAStorage. As described above, the LTAStorage has its own logic to retrieve a file. It searches the FileObject table, and obtains a list of fileobjects which are associated with DataObject `data_obj`. It sorts the list of fileobjects, the first fileobject in the list is retrieved using the corresponding protocol, in this case, the GridProtocol. The retrieved file is then returned to the DataObject.

The advantage of using this framework is that it can easily be extended to support other storage systems. What we need to do is to add the corresponding protocol interface of the storage system, which will be used to get and put data from or to the storage.

5 Applications

In the following section we will review three different applications on Astro-WISE as typical examples: KIDS data processing as a type of application which uses EGEE and Astro-WISE computing facilities and Astro-WISE storage facilities, LOFAR Long Term Archive which uses both Astro-WISE and EGEE for data processing and data storage. The Monk application is an example of the first non-astronomical data model realized in Astro-WISE which can use the computing facilities of EGEE as well.

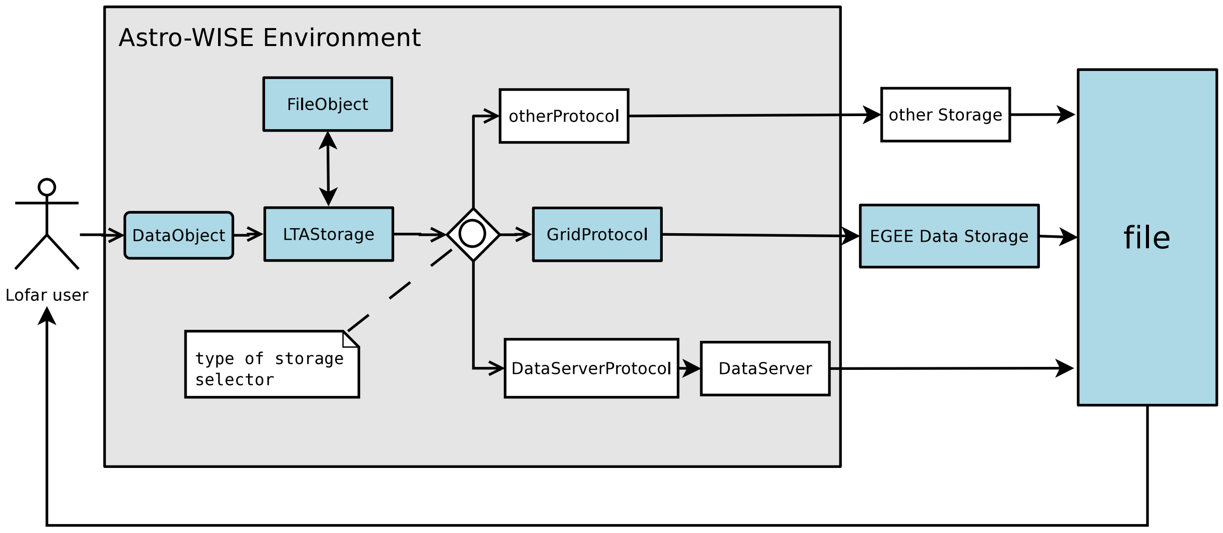


Fig. 10 Access to data files on Astro-WISE with an implementation of EGEE storage

5.1 KIDS

The Kilo-Degree Survey (KIDS) mentioned in Section 2.1 is a collection of images of 1° × 1° size. From the data processing point of view each such a field is a float array of 268,435,456 elements. For each of roughly 50,000 image we have to go through a number of steps to the final catalog, producing and storing all intermediate data products, which include new images, data processing constants and parameters of data processing.

Figure 11 shows a hierarchy of all classes used in the data processing. This hierarchy of classes is a data model for KIDS data processing. It was implemented as a hierarchy of classes in Python libraries of Astro-WISE and as a set of tables in metadata database.

To get from the top of Fig. 11 to the bottom requires a number of processing steps, which involves submitting a job to one of the Astro-WISE computing elements (Fig. 12). The user can select the computing element and the type of

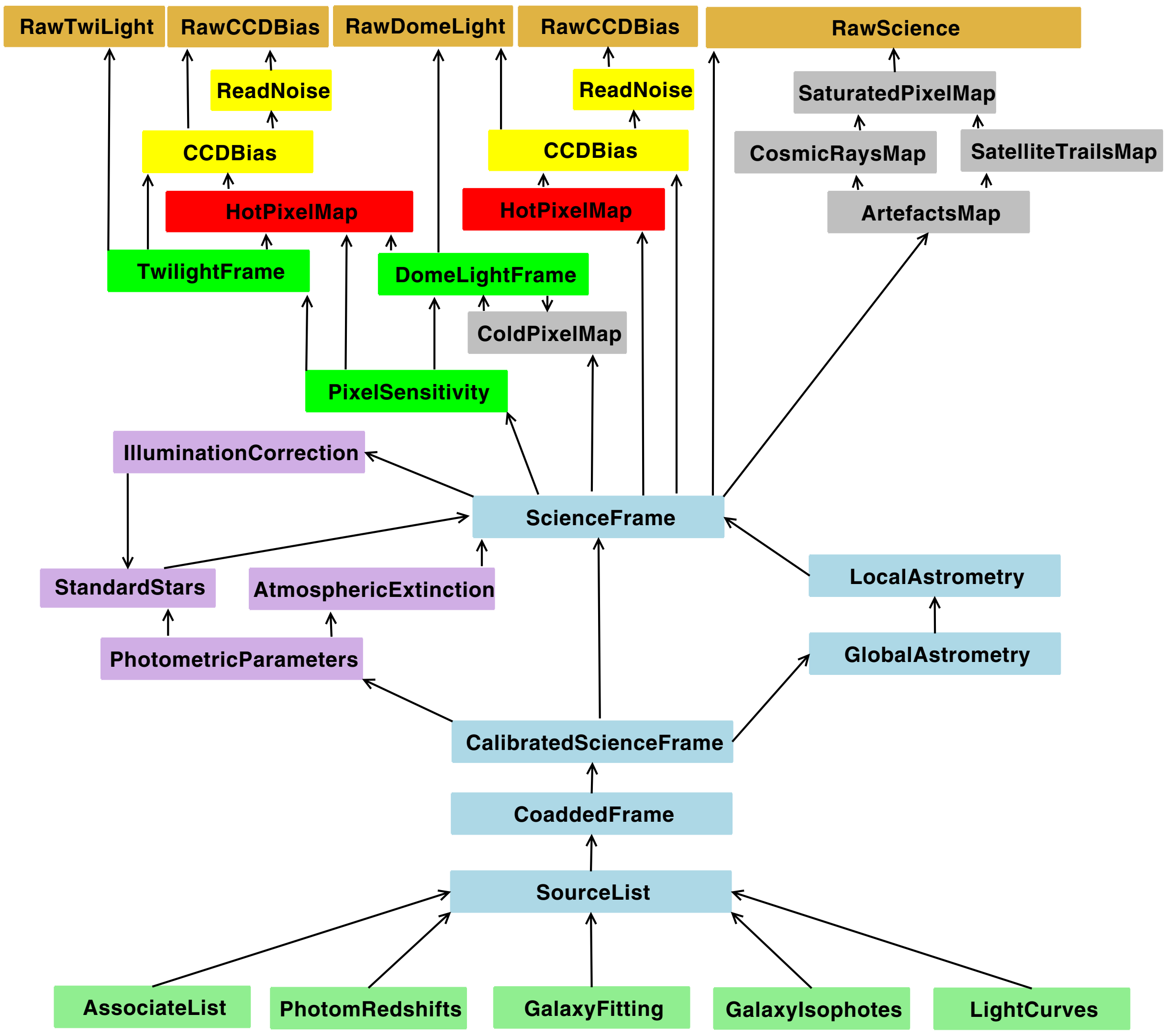
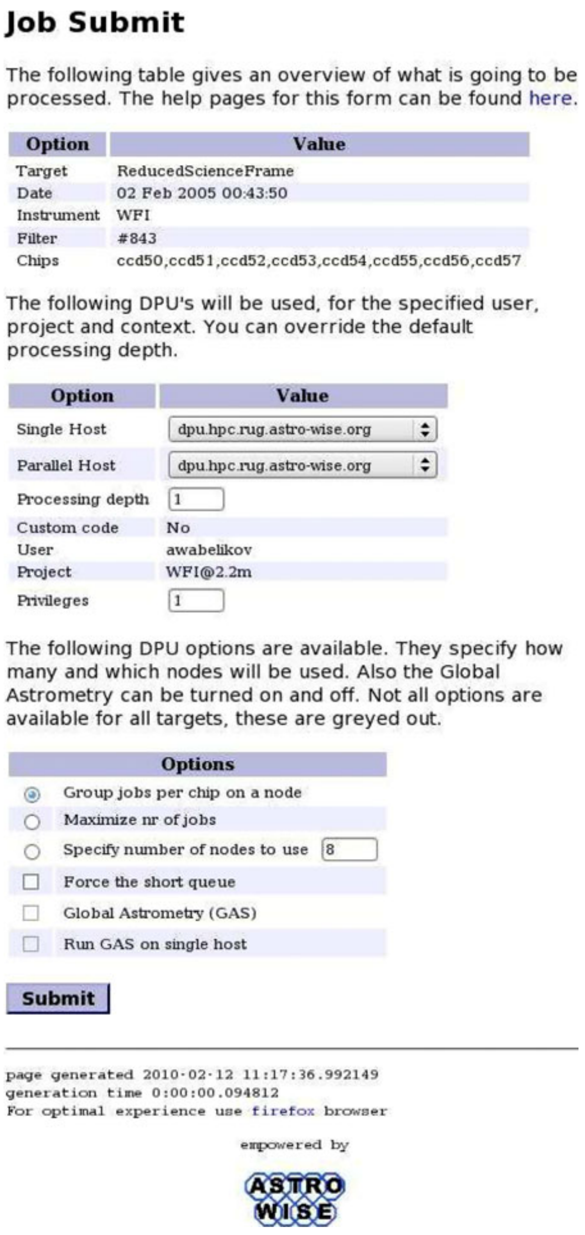


Fig. 11 An optical image data model implemented in Astro-WISE for optical image processing (KIDS)



Job Submit

The following table gives an overview of what is going to be processed. The help pages for this form can be found here.

Option	Value
Target	ReducedScienceFrame
Date	02 Feb 2005 00:43:50
Instrument	WFI
Filter	#843
Chips	ccd50,ccd51,ccd52,ccd53,ccd54,ccd55,ccd56,ccd57

The following DPU's will be used, for the specified user, project and context. You can override the default processing depth.

Option	Value
Single Host	dpu.hpc.rug.astro-wise.org
Parallel Host	dpu.hpc.rug.astro-wise.org
Processing depth	1
Custom code	No
User	awabelikov
Project	WFI@2.2m
Privileges	1

The following DPU options are available. They specify how many and which nodes will be used. Also the Global Astrometry can be turned on and off. Not all options are available for all targets, these are greyed out.

Options
● Group jobs per chip on a node
○ Maximize nr of jobs
○ Specify number of nodes to use 8
☐ Force the short queue
☐ Global Astrometry (GAS)
☐ Run GAS on single host

Submit

```
page generated 2010-02-12 11:17:36.992149
generation time 0:00:00.094812
For optimal experience use firefox browser
```

empowered by

ASTRO WISE

Fig. 12 Target processing in Astro-WISE. Selection of the processing parameters

data processing (single node processing or parallel jobs, computing element). The job will create new objects which will be ingested into the system. They can be seen and processed further with the use of this interface or other interfaces described in Section 2.6. The use of an EGEE computing element requires the registration of users within Virtual Organization (*omegac* for Astro-WISE, *lofar* for LOFAR) and the use of the MyProxy tool (see Section 4.2 for the detailed description). More on the astronomical image processing with Astro-WISE can be learned from the tutorials on the Astro-WISE site.[43]

5.2 LOFAR Long Term Archive

LOFAR, the LOw Frequency ARray, is building a huge radio telescope.[44] In the first building phase, LOFAR will consist of a compact core area (approx. 2 km in diameter, 32 stations) and 45 remote stations. Each station will be equipped with 100 high band antennas and 100 low band antennas. Each station is connected to the Central Processing Unit by a Wide Area Network (WAN) with 10 Gbps net.

The LOFAR infrastructure performs an aperture synthesis at the Central Processing Unit based on the signals from each station. The Central Processing Unit is using the BlueGene/P supercomputer from IBM and PC clusters at the University of Groningen.

Thanks to its unprecedented sensitivity, two LOFAR projects can explore the end of the early stages of the Universe. The LOFAR Surveys project will identify extremely distant radio galaxies using an empirical correlation between the radio spectral steepness and distance. This first time inventory of extremely high redshift radio galaxies will constrain the formation of supermassive black holes (the radio emission is powered by those black holes), yield a detailed view of the interstellar medium (the fuel of star formation which shows up as absorption of the radio emission) and identify proto-clusters, as distant radio galaxies have been shown to pinpoint those.

The main challenge of LOFAR is the data storage which will exceed 20 PB for the first 5 years of observations. Because both the research groups participating in LOFAR and the available storage and computing resources are distributed, the data must be stored in different geographical locations,

[43]http://www.astro-wise.org/portal/aw_howtos.shtml

[44]http://www.lofar.nl

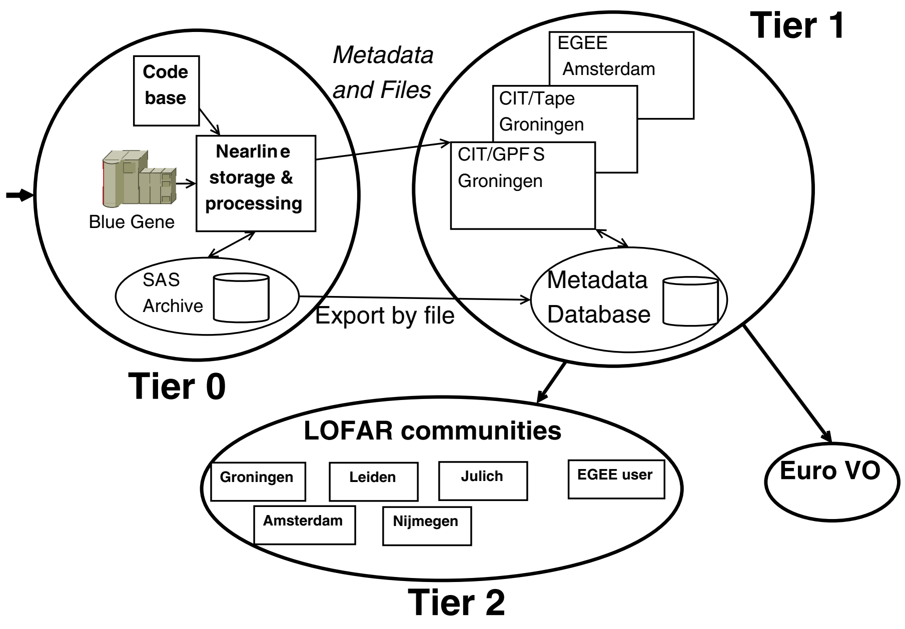


Fig. 13 Tier architecture adopted by Astro-WISE for LOFAR Long Term Archive

while being accessible for retrieval and reprocessing by any LOFAR user.

The LOFAR Information System, developed as an extension of Astro-WISE, will deal with the data delivered by the Central Processing Unit. This data is stored in the Long Term Archive (LTA). The LTA is the main driver for the merging of Astro-WISE and EGEE, as many groups of scientists will store and reprocess their data on EGEE, but still want to use a single logically centralized data storage repository with the support for intensive data mining.

The LOFAR architecture is described in details in [17]. The data of all stations is collected by Central Processing Unit (CEP) and processed with the use of Blue Gene/P (BG/P) with 34 TFlops peak processing power. The results of the processing are stored in the Long Term Archive.

The scale of the data storage and computing for LOFAR reaches the level of the Large Hadron Collider experiment (LHC), and as in the case of LHC/EGEE [18] we adopted a tier architecture for the LOFAR data storage and data processing Grid (see Fig. 13). In the case of LOFAR we have both similarities with the EGEE architecture ("one-way" data transfer from Tier 0 to Tier 1) and differences (no data will be stored on Tier 2 level, apart from the end-user private data).

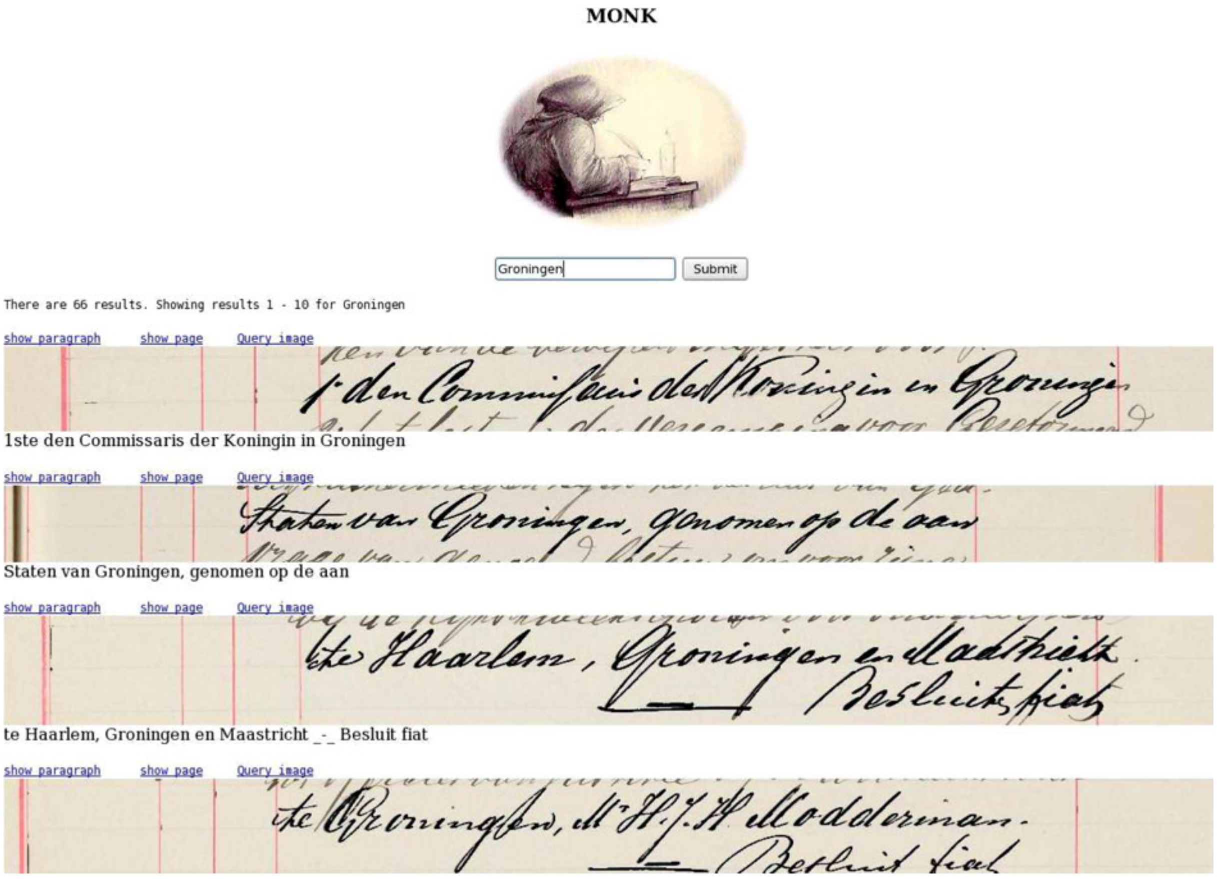
Tier 0 is the Central Processing Unit which receives information, provides necessary computations, and delivers the data to Tier 1. Tier 2 is an end-point for the user access to the data and computational facilities.

Tier 1 consists of a number of data storage nodes and computing nodes deployed in Groningen (Donald Smits Center for Information Technology, CIT[45]) or at any location where the research group participating in the project can provide a facility for data storage. Nodes of Tier 1 can be an Astro-WISE dataserver, a global filesystem node (GPFS) or an EGEE node.

Users can operate from nodes of Tier 2 to browse the metadata database (which is stored on Tier 1 nodes), to start computations on computing elements of Tier 1 and to retrieve, update or insert data on Tier 1.

The system is based on the existing Astro-WISE information system and will inherit its main features and abilities. The data will be stored both on Astro-WISE dataservers and on EGEE storage nodes. Astro-WISE is already implementing the "step-in model" of the LOFAR Long Term Archive, the testing ground and the prototype of

[45]http://www.rug.nl/cit


MONK

Groningen Submit

There are 66 results. Showing results 1 - 10 for Groningen

show paragraph show page Query image

1ste den Commissaris der Koningin in Groningen

show paragraph show page Query image

Staten van Groningen, genomen op de aan

show paragraph show page Query image

te Haarlem, Groningen en Maastricht _-_ Besluit fiat

show paragraph show page Query image

Fig. 14 Monk interface for the search of the word pattern in hand-written documents, the search word is "Groningen", the extracted lines of the documents with this word are shown

LTA. It contains 200 Tbytes of LOFAR data. The description of the LOFAR Information System design is done in [22].

5.3 Text Recognition

During the last 2 years, the AI department of the University of Groningen[46] achieved a significant progress in the datamining of handwritten collections. The availability of high-performance computing facilities has made it possible to explore new concepts in machine learning and pattern recognition. By combining brute force (correlator) matching with traditional machine learning using Hidden–Markov Models and Support Vector Machines, it was proved that continuous web-based transcription of text, concurrent with nightly retraining computation sessions allows to open up access to large collections of scans of handwritten text. The description of the methods behind this work can be found in details in [19] and [20].

However, this success has caused a snowball effect in terms of the requirements for a continuous availability of high-performance computing, persistent storage of valuable cultural-heritage images for several decades, and a fusion of modern web-based data access for both cultural-heritage researchers and computer scientists working on pattern recognition and machine-learning problems. A pilot experiment performed

[46]http://www.rug.nl/ai

in collaboration with the Astro-WISE staff has demonstrated that much can be gained from joint access to a central data-operations and science-operations center. Given the continuously available computing power and the reliable storage of raw data, systematic experiments can now be planned using the Astro-WISE platform. Web-based access to experimental results for pattern recognizers is functionally not very different from similar data and image access by astronomers. However, in order to maximally benefit from a compute/store service, a new data model has been implemented in Astro-WISE for the text recognition task.

Having scanned one book, i.e., the year 1903 of the "Cabinet of the Queen" at the Dutch National Archive, it now becomes a scientific and technical challenge to scale up towards providing access to the full range of years from 1901 to 1915. The general applicability of the approach and the generalizability of the classification performance of the classifiers used are becoming "hot" research questions. How good are our models in the prediction of handwritten shapes, forward and backward in time? Apart from scientific output in the area of pattern recognition and technical output in terms of an architecture supporting a science-operations center, there will be the tremendously satisfactory result of allowing the general public access to these handwritten collections via a search engine.

The work done in the collaboration with the Artificial Intelligence department of the University of Groningen is an example of a realization of a new, non-astronomical data model in Astro-WISE. Also a web service called Monk[47] was developed to search the recognized text and show the original scans (see Fig. 14).

6 Conclusion and Future Work

The nearest future will bring a number of applications using the Astro-WISE information system and its clones, apart from the three described above. In the North of the Netherlands a large multi-disciplinary initiative, named TARGET,[48] has been started to further deploy the typical data-centric Astro-WISE approach, integrated with Grid computing, for a wide variety of research and business programs. The Target program is a collaboration between industry (IBM, Oracle, Target Holding) and University of Groningen, Astron[49] and the Academic Hospital (UMCG[50]).

For the UMCG Lifelines project,[51] genome and phenotype data of 165,000 patients are gathered over a period over 35 years and brought in an system based on Astro-WISE (Life-WISE[52]). Synergies with industrial products such as Oracle Life-science hub are remarkable and will be further explored.

The Target project focuses on research communities working with extremely large data sets, with ever changing code/methods, typical for a research environment. The need for full data lineage and compliance with an ISO Open Archival Information System Reference Model[53] is common for all the programs in the Target project.

The merging of Astro-WISE and EGEE Grid extended the capabilities of Astro-WISE. At the same time the implementation of the customized data model in Astro-WISE allows to enhance the abilities of EGEE to organize a complicated data processing and to store metadata of the data items. Apart from porting of Astro-WISE applications on EGEE and implementing data storage for the LOFAR Long Term Archive this work gave a chance to study the possibility to merge two different concepts of Grid and revealed the problems of this integration.

One of the problems which we are going to address in the future is the handling of failures in job submission and the access to the data storage. The handling of job submission failures should include proper classification of the failure and the implementation of policies in the case of the

[47]http://www.ai.rug.nl/Monk

[48]http://www.rug.nl/target

[49]http://www.astron.nl

[50]http://www.umcg.nl

[51]http://www.lifelines.net

[52]http://www.rug.nl/target/partners/lifelines

[53]http://public.ccsds.org/publications/archive/650x0b1.pdf

failure. Similar problems should be addressed in the case of data storage failure (for example, lost connection with designated storage element).

Another problem is that most of the current jobs in Astro-WISE system are run in parallel. The use of a parallel job type limits the number of EGEE computing elements where Astro-WISE can be run, because the support for this is currently not widely implemented. It also depends on a reliable way to start up the tasks on the assigned compute nodes. It is therefore worth investigating if single jobs can be used instead. This will mean, however, that synchronisation of the jobs will be an issue. If no synchronisation is applied, but a new series of jobs is started instead, the amount of data transport to and from the compute nodes will become much higher.

Another issue that we may encounter is a high failure rate for submitted jobs, due to the large scale and complexity of the Grid infrastructure. The use of pilot jobs, where jobs connect back to a central service for obtaining a number of workloads is widely spread. This helps solving or, better said, hiding the problem from the user, because jobs that fail will not connect back, and the successful jobs will handle all the tasks. Since Astro-WISE already uses a model where the jobs connect to the central system for a workload, extending this to be able to hide problematic nodes should not be very difficult.

Finally, we are going to perform a series of tests to compare the performance of EGEE and Astro-WISE computing nodes on different types of jobs for a better selection of the computing node for each job.

Acknowledgements Astro-WISE is an on-going project which started from a FP5 RTD programme funded by the EC Action "Enhancing Access to Research Infrastructures". This work is supported by FP7 specific programme "Capacities—Optimising the use and development of research infrastructures" and is done in the framework of EGEE-III NA4 Astronomy & Astrophysics Cluster. We are grateful to Dr. J. McFarland (Kapteyn Astronomical Institute, Groningen University) for valuable comments on this paper. This work used the EGEE[54] infrastructure, cofunded by the European Commission (INFSO-RI-222667).

[54]http://www.eu-egee.org

References

1. Valentijn, E., et al.: Astro-WISE: chaining to the universe. In: Shaw, R.A., Hill, F., Bell, D.J. (eds.) Proc. of ADASS XVI, ASP Conf. Ser., vol. 376, 491 (2007)
2. Taffoni, G., et al.: Enabling Grid technologies for Planck space mission. Future Gener. Comput. Syst. **23**, 129 (2007)
3. Boku, T., et al.: HMCS-G: Grid-enabled hybrid computing system for computational astrophysics. Third IEEE international symposium on cluster computing and the Grid (CCGrid'03), p. 558 (2003)
4. Ibarra, A., et al.: On-the-fly XMM-Newton spacecraft data reduction on the Grid. Sci. Program. **14**, 141 (2006)
5. Taffoni, G., et al.: Bridging the virtual observatory and the Grid with the query element. Grid Workshop, astro-ph/0605165 (2006)
6. Jones, B.: An overview of the EGEE project. In: Book Series Lecture Notes in Computer Science, vol. 3664, 1 (2005)
7. Laure, E., et al.: Programming the Grid with gLite. Comput. Methods Sci. Technol. **12**, 33 (2006)
8. Pordes, R., et al.: The open science Grid. J. Phys. Conf. Ser. **78**, 012057 (2007)
9. Foster, I.: Globus toolkit version 4: software for service-oriented systems. In: IFIP International Conference on Network and Parallel Computing. LNCS, vol. 3779, p. 2. Springer, Berlin (2006)
10. Thain, D., Tannenbaum, T., Livny, M.: Condor and the Grid. In: Berman, F., Hey, A.J.G., Fox, G. (eds.) Grid Computing: Making The Global Infrastructure a Reality. Wiley, New York (2003); Condor High-Throughput Computing Software Framework
11. Nisson, P., et al.: The PanDA system in the Atlas experiment. In: Speer, T., Carminati, F., Werlen, M. (eds.) Proceedings of XII Advanced Computing and Analysis Techniques in Physics Research, Erice, Italy 1, (2008)
12. Moscicki, J.T., et al.: GANGA: a tool for computational-task management and easy access to Grid resources. Comput. Phys. Commun. **180**, 2303 (2009)
13. Koblitz, B., Santos, N., Pose, V.: The AMGA metadata service. J. Grid Comput. **6**, 61 (2008)
14. Alfieri, R., et al.: From Gridmap-file to VOMS: managing authorization in a Grid environment. Future Gener. Comput. Syst. **21**, 549 (2005)
15. Lawrence, B.N., Kershaw, P., Blower, J.: Practical access control with NDG-security. In: Cox, S. (ed.) Proc. UK e-Science All Hands Meeting (2007)

16. Latham, S.E., et al.: The NERC Data Grid services. Phil. Trans. R. Soc. A **367**, 1015 (2009)
17. Gunst, A.W., Bentum, M.J.: Signal processing aspects of the low frequency array. In: Proc. ICSPC 2007, 600 (2007)
18. Stewart, G.A., Cameron, D., Cowan, G.A., McCance, G.: Storage and data management in EGEE. In: Proc. Fifth Australasian Symposium on Grid Computing and e-Research (AusGrid 2007). CRPIT, Ballarat, Australia, p. 68 (2007)
19. Van der Zant, T., Schomaker, L.R.B., Haak, K.: Handwritten-word spotting using biologically inspired features. IEEE Trans. Pattern Anal. Mach. Intell. **30**(11), 1945 (2008)
20. Schomaker, L.R.B.: Word mining in a sparsely-labeled handwritten collection. In: Yanikoglu, B.A., Berkner, K. (eds.) Proceedings of Document Recognition and Retrieval XV, IS&T/SPIE International Symposium on Electronic Imaging, pp. 6815–6823 (2008)
21. Foster, I.: What is the Grid? A Three Point Checklist, GRIDToday (2002)
22. Valentijn, E., Belikov, A.: LOFAR information system design. Mem. Soc. Astron. Ital. **80**, 509 (2009)
23. Mwebase, J., Boxhoorn, D., Valentijn, E.: Astro-WISE: Tracing and Using Lineage for Scientific Data Processing. Conference on Network-Based Information Systems, 475 (2009)