The VO-Dance web application at the IA2 data center

Marco Molinaro, Cristina Knapic and Riccardo Smareglia

Istituto Nazionale di Astrofisica - Osservatorio Astronomico di Trieste, via G.B. Tiepolo 11, Trieste, Italy

ABSTRACT

Italian center for Astronomical Archives (IA2, http://ia2.oats.inaf.it) is a national infrastructure project of the Italian National Institute for Astrophysics (*Istituto Nazionale di AstroFisica*, INAF) that provides services for the astronomical community. Besides data hosting for the Large Binocular Telescope (LBT) Corporation, the Galileo National Telescope (*Telescopio Nazionale Galileo*, TNG) Consortium and other telescopes and instruments, IA2 offers proprietary and public data access through user portals (both developed and mirrored) and deploys resources complying the Virtual Observatory (VO) standards. Archiving systems and web interfaces are developed to be extremely flexible about adding new instruments from other telescopes. VO resources publishing, along with data access portals, implements the International Virtual Observatory Alliance (IVOA) protocols providing astronomers with new ways of analyzing data. Given the large variety of data flavours and IVOA standards, the need for tools to easily accomplish data ingestion and data publishing arises. This paper describes the VO-Dance tool, that IA2 started developing to address VO resources publishing in a dynamical way from already existent database tables or views. The tool consists in a Java web application, potentially DBMS and platform independent, that stores internally the services' metadata and information, exposes restful endpoints to accept VO queries for these services and dynamically translates calls to these endpoints to SQL queries coherent with the published table or view. In response to the call VO-Dance translates back the database answer in a VO compliant way.

Keywords: VO, archive, resource publishing, data ingestion

1. INTRODUCTION

IA2, a data center hosted by the INAF-Trieste Astronomical Observatory (*INAF-Osservatorio Astronomico di Trieste*, INAF-OATs), is a national infrastructure project that has as its long term goal *'to implement a new strategy for preserving and providing access to the Astrophysical data heritage'* (see *http://ia2.oats.inaf.it*). To achieve this result the development of flexible an re-usable tools is needed for the common operations a data center requires, from data archiving to resource deployment for the end user.

In this paper we present the IA2 data center (Sec. 2) describing the data it manages, the user portals it exposes and the applications used for data ingestion. The Distributed Archiving System (DAS, Sec. 2.2), responsible for data managment and distribution will be sketched out. We will then briefly describe the VO world (Sec. 3) and the access protocols defined by the IVOA to standardize data access. After this descriptive *scenario* we will introduce the VO-Dance tool (Sec. 4) and how it translates the archived data into VO compliant format. VO-Dance description will be splitted into its tokens (via architecture description, VO metadata characterization and actual service publishing steps) to allow better insight of the application and its requirements (see Sections 4.1 through 4.3). Finally, we will draw some conclusions on the state-of-the-art at IA2 and VO-Dance and what will be the next planned improvements.

Contact author: Marco Molinaro, e-mail: molinaro@oats.inaf.it, tel.: +39 040 3199 152

Software and Cyberinfrastructure for Astronomy II, edited by Nicole M. Radziwill, Gianluca Chiozzi, Proc. of SPIE Vol. 8451, 845105 · © 2012 SPIE · CCC code: 0277-786X/12/$18 · doi: 10.1117/12.924816

2. IA2

IA2 is an ambitious italian research infrastructure project that aims at co-ordinating different national initiatives to improve the quality of astrophysical data services for research purposes. IA2 developes and maintains the Distributed Archiving System (DAS) for International Telescope Corporations and National Telescopes located around the world. LBT, TNG and Asiago Astronomical Observatory (*Osservatorio Astronomico di Asiago*, OAA) data are fully managed by IA2 starting from instrumental product raw data stored in FITS (Flexible Image Transport System) format till their release to the astronomical community through dedicated user interfaces and IVOA compliant web services.

Besides these fully managed instruments, IA2 hosts mirrors of web portals from other astrophysical projects like REM (Rapid Eye Mount), data from other projects and also services and tools for stellar and cosmological simulated data projects (BaSTI, a Bag of Stellar Isochrones and Tracks, and a part of the Italian Theoretical Virtual Observatory, ITVO).

At present it is also involved in the starting phases of the CTA (Cherenkov Telescope Array) project, in the publishing efforts of the Gaia mission, in giving support to the GAPS (Global Architecture of Planetary Systems) project for the HARPS-N high-resolution spectrograph for the TNG telescope and to the PESSTO (Public ESO Spectroscopic Survey for Transient Objects) survey.

Hereafter we detail the managed and hosted data and services, while in Section 2.2 we describe the Distributed Archiving System.

2.1 Data archives and services

IA2 started from the TNG public data (see e.g. [1]) by exploiting the experience acquired through the construction of the TNG-LTA (TNG-Long Term Archive) prototype archive. The TNG-LTA pilot project has made possible to study and to define an optimal structure as regards scientific raw data and housekeeping telemetry, together with their processing; furthermore, a big evolution concerned the interoperability concept. This meant harmonization of the LTA to current IVOA standards thus allowing data access not only through a dedicated portal including both public and proprietary data but also via VO dedicated services (see e.g. Section 4.4). At present all data from TNG (public and private) are handled at the IA2 data center.

For the LBT-DA (LBT-Distributed Archive [2]) IA2 manages the archive infrastructure for the raw science and calibration exposures of all LBT instruments. The Distributed Archive consists of 4 mirror sites (for details see Sec. 2.2) that collect national data; the LBT-DA mirror archive in Trieste, for example, collects all data acquired during the observing time assigned to Italy. In addition all mirror sites archive the whole metadata. As for TNG, also for LBT the data center provides publishing services along with ingestion and archival tasks. Publishing consists in dedicated public/proprietary data portal for data retrieval and a VO compliant image web service.

Recently (roughly from December 2011) IA2 manages also the data for the Asiago Observatory. Its telescopes and instruments are operated by the INAF-Padua Astronomical Observatory while the data center (hosted by INAF-OATs) provides ingestion, archiving and publishing systems with tools directly derived or already available from the TNG and LBT experience.

The data center portal hosts also a mirror of the REM archive portal.

Besides observational data archiving and distribution, IA2 is also involved in simulated data projects. The main one is the ITVO (see [3] and [4] for further details), a multi-site archive for theoretical cosmological data. ITVO is an effort of VObs.it, the italian effort for the VO part of the EURO-VO project and directly involved in the IVOA. The INAF-OATs ITVO site consists of an archived cosmological simulation of galaxy clusters and a web portal that fully exposes that simulation on various levels, from simple data retrieval through 2D and 3D on demand visualization, to metadata previewing. All of these tasks performed using VO tools whenever possible to obtain interoperable services and results. The database structure, the publishing portal and the visualization applications provided through web services have been all developed at IA2 [5].

BaSTI (also part of ITVO) is a project of the INAF-Teramo Astrophysical Observatory providing simulated stellar evolutionary data (for details see [6]). The collaboration with IA2 at INAF-OATs for this project consisted

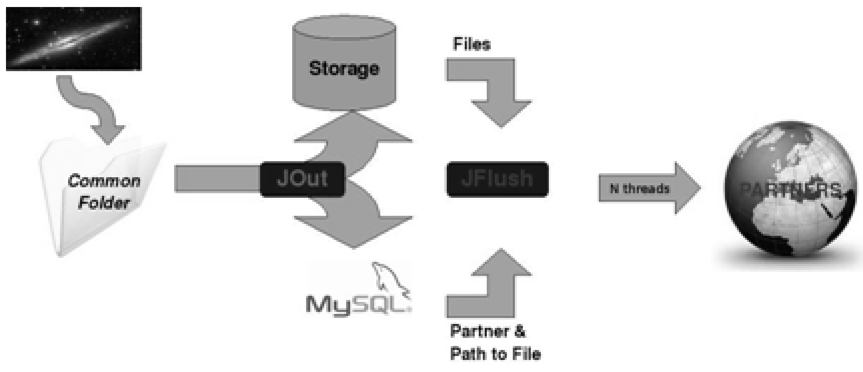

Figure 1. DAS workflow.
Astronomical data inserted in a common folder is detected by a system utility and managed by JOut. It provides insertion of metadata in MySQL database, while data are stored in a high availability storage cluster for preservation. Once archived, JFlush checks the DB for data to be sent to recipients and distributes them in a multithread way to the Partners all around the world.

in providing a reliable database to allow simple searches for those simulation files and let the project grow up. IA2 provided also a portal for the project with web services (server side applications) to preview and manipulate the simulated data itself. Again this was done taking advantage of VO existing tools and libraries.

As an effort to allow data preservation and accessibility IA2 hosts also data from other projects. One of them is the UVES Large Program (ULP) for testing fundamental physics, an ESO proposed application by P.Molaro. Another is the Architecture and Tomography of Galaxy Clusters (ATCG) project based upon an ESO Large Program led by P. Rosati and co-ordinated in Italy by M. Nonino. For the ATCG project one deliverable is the data release through VO services taking advantage of the VO-Dance application.

The IA2 data center is a growing infrastructure and this arises the need of archiving, ingestion and publication tools independent and easily adaptable in case of new instrument or project addition. The DAS (next Section) and VO-Dance (Section 4) tools are developed to try to simplify these tasks.

2.2 Distributed Archiving System

The DAS was designed and developed to manage, maintain and distribute LBT data products between four main sites located in Arizona (Mount Graham and Tucson, U.S.A.), Heidelberg (Germany) and Trieste (Italy). DAS replicates metadata at every archive site but allowing file dissemination consonant to the data ownership and policy. It is also possible to use DAS for non distributed archives, as is the case of TNG and OAA, thus creating a data base for the publication of services from calibrated data (e.g. through VO-Dance, see Sec. 4.4). Here we briefly describe its functionality (refer also to Fig. 1).

Currently DAS requires that the input files are valid FITS containing some specific header keywords describing the celestial coordinates of the image/data, the name of the instrument, the acquisition date and data ownership. Given these constrains, FITS header's metadata are read and coded with a dedicated software written in Java (the *db_creator*) that, starting from an existing database schema (one MySQL DBMS schema for each telescope DAS is configured for), creates technical, instrumental and general purpose tables. The *db_creator* implements also a technical table (named *key2db*) that stores information and special settings to choose the keywords to be inserted as metadata into general purpose and instrument dedicated tables. The *key2db* table is the main configuration system of the DAS and is used also to provide consistency (through a dedicated database procedure) between metadata types and database columns type definitions.

When new data is to be ingested in the DAS managed system, the data is temporarily stored in a common folder dedicated to each telescope. This folder is monitored for new data files by a Java system utility (*JNotify*) that starts the file management process (another Java application: *JOut*). *JOut*, following the directives stored in the *key2db* table, inserts a subset of the FITS keywords in an instrument dedicated schema table and in a general table common to all the data (of the telescope). The technical table contains information like file location, file status, transfer file status, publication policy and so on. Instrumental tables contain all relevant metadata stored in FITS headers keywords. Some operations are performed on metadata using MySQL procedures to adjust eventually forgotten keywords or null values and to synchronize the instrument dedicated table with the general purpose table. When all relevant information is correctly retrieved from the FITS file a new record is inserted in the database, the corresponding file is copied/moved in the archive (on a dedicated storage) and copied on a temporary user dedicated folder directly accessible from local computers. Files are usually stored in compressed format. If errors occurs, alternative proceduress are performed, depending on error type, but file preservation has top priority. When data are correctly inserted in the database, the files are archived and the updates are performed, the file is ready to be sent to the other recipients (corporation partners and owners) throught the use of *JFlush*, a dedicated software for delivering. *JFlush* makes a deep use of multi-threading Java technology to read the URL of a recipient from the technical table in the database and collect a list of files to be delivered to it using the blocking queue mechanism. A specified number of threads are in charge to retrieve files from the archive and send them properly to destination URL, checking for delivery success, and put them on the recipient's server. Once a thread finishes its job, another thread takes its place mantaining the total number of active threads, till the end of the queue. Multi-threading technology is used in order to guarantee the maximum possible transfer rate, optimizing band occupancy.

At all archive sites a user interface for data retrieval is deployed, allowing for query on all telescope data or more specific queries on single instruments. Authorized users can also download single files, tar archives (for more files) or a single table describing the current result of a query in VO compliant format (VOTable [7]). Indeed, the user interface result display, and the consequent query result, are formed depending on the user account role (admin, public, partner, PI, program, . . .). Besides prioprietary data, release of public data is dependant on partner's policy but, generally, all metadata are accessible and issued in form of simple VOTables.

Currently, files, tar files and VOTables are served directly as results of the user interface process but future plans are to use a *FileServer*, a web service that works as an application-indipendent validator for the incoming file requests. A FileServer prototype is already in production use to serve public data requests by the VO clients that call the IA2 VO published services (see Section 4 for further details on these services).

3. VIRTUAL OBSERVATORY

DAS and file serving (whichever the serving solution) provides a robust way to serve metadata and data, especially when providing a user interface, because it means direct data access through the world wide web. However, when dealing with different data flavours, multiple telescopes, instruments or projects, with metadata content changes, it could be difficult to afford, in a fast and reliable way, the needed service changes. A solution to this scenario is what we will be describing at Section 4 with VO-Dance, but before describing it we will briefly describe the VO world and the standards it provides that are connected to the VO-Dance development.

The IVOA (*http://www.ivoa.net*) is an alliance of national or super-national projects and institutions that are trying to develop a common web infrastructure to allow astrophysical data distribution. The idea can be compared to that of the world wide web, that is a (large) set of resources that can be queried, described within a common set of standards and protocols and, most important, are interoperable and transparent to the web users. The same way the VO is the idea of deploying astrophysical resources on the common basis of standards and protocols (provided by the IVOA) to let the end users (the astrophysical community) find the data they are searching for in a transparent way using only web browser or VO-aware clients. This means the astronomer can access, for example, optical images of some deep sky object in a given waveband without needing to know where the images are stored nor needing to understand VO standards or protocols. The user will only need to have a proper client and that the data be public.

This is really a short description of the VO idea, efforts are going on to provide it with credential capabilities to allow also proprietary data to be reached directly through VO applications (see e.g. in conclusions regarding Single Sign On efforts). Also, standards and protocols are improving, because the VO is a relatively young effort within the world astrophysical community. We will now shortly describe some of the protocols that define common access to astrophysical data and can be used to deploy VO compliant resources.

3.1 VO standards and protocols

Protocols and standards shown in the following subsections give a basis for the description of the VO-Dance application on next section (Sec. 4). These descriptions are not intended to be a complete protocol reference nor a full list of IVOA standards and protocols, that can be found at the IVOA website under the *Documents* branch (see *http://www.ivoa.net/Documents*). We will firstly describe some concepts (content descriptors and universal types) that are widely used in the VO and its protocols.

3.1.1 Content descriptors and types

The output of VO services generically consist in sets of resources or items listed as table rows. The preferred format for that table is a, so called, VOTable, a format defined using the XML language. To identify and describe the tagged values that form this XML table output to the clients that obtain it from the queried service, various attributes have been standardized (or are under standardization). These attributes are needed to characterize fields otherwise identified only by a column name, they provide information in a way similar the FITS headers provide insight on FITS data content but with finer information.

One set of descriptors is formed by the Unified Content Descriptors (UCD [8]) that are used to identify the meaning of the field, e.g. the field containing the right ascension for a record will have a UCD attribute set to a unique text string to identify it. Togheter with UCDs, to express the field meaning related to a particular data model used, UTypes can be provided. The *unique* or *uniform* types (UTypes) are not yet an IVOA standard, but are widely used in the VO to uniquely define resources, resource's tokens, fields. Their usage is currently defined by the standard protocols (see e.g. the SSAP, Section 3.1.4 hereafter). They can be seen having the same meaning of an RDF vocabulary for VO models. The protocol for spectral data access (see subsection 3.1.4) makes wide use of UTypes for identify field requirements.

Togheter with UCDs and UTypes other metadata can characterize the data exposed through VO protocols giving VO clients more information on how to interpret data files and improving interoperability between datasets and tools. One of them is, for example, Units (that is an IVOA recommandation in progress): exactly as the name suggests, these provide information on the units of the field they refer to. Their aim is to allow automatic conversion for the same field exposed with different unit values.

All of these metadata characterization, devoted to data modelling and interoperability, concur at the IVOA aim of creating a publishing layer between data providers and astrophysical community that is completely transparent to the end user while allowing for data retrieval and analysis using data with common meaning but heterogeneous formats and archiving solutions.

The protocols we will sketchily describe in the next four subsections make use of these descriptors and allow data access being different only on the type of data one searches for: catalogues, images, spectra or generic datasets.

3.1.2 Cone Search

This protocol is defined to provide a standard way to access tabular catalogues with positional parameters. It only applies to single table catalogues and the required query parameters are right ascension (RA) and declination (DEC) positional parameters and a search radius (SR). This means records in a Cone Search published resource are retrieved when the position they refer to falls inside a cone centered at RA,DEC and of SR angular size. All oher catalogue fields can or cannot be used to search the catalogue depending on how the resource is deployed. Similarly, all the other catalogue fields can be returned in response by the resource depending on its configuration. Of corse usually all available fields, if not too many, are returned, because a three column table (RA, DEC and an identifier, this last required on response) would be a meaningless result. This protocol is easy to follow and comply to (requires only basic information characterized by the correct UCDs) but has some clear disadvantages on search capabilities, this is why a different multi-table protocol has been later developed, the Tabular Acces Protocol (TAP) that will be described in section 3.1.5 hereafter.

3.1.3 Simple Image Access

Another widely used protocol is the one devoted to image access, thus called Simple Image Access Protocol (SIAP [9]). The constraints for SIAP compliance are more complex than those of a Cone Search, requiring image information (image dimensions, spatial resolution, format and image file access reference mainly) other than the simple position of the target. SIAP, moreover, provides a wide set of values and informations that are not strictly required but are strongly suggested to be returned by the service. World Coordinate System (WCS, see e.g. *http://www.atnf.csiro.au/people/mcalabre/WCS/* for further details) characterization, exposure and time of observation, estimated file dimensions, are some of the fields that can better qualify a query response from a SIAP compliant service. The basic service constraints are however quite simple and then also a SIAP service is not a too much difficult one to deploy. Actually one is asked to provide a service capable of understanding a RA and DEC positional value (in this protocol these two values are put togheter in one HTTP query parameter named POS), plus a SIZE parameter composed of two values, to outline a rectangular-like box on the celestial sphere, and to return a VOTable containing the information sketchily described here above for each record that meets the query. One of the required fields of the table (the URI of the file that contains the image) will provide the client a way to retrieve it. This means that the SIAP response doesn't contain actual data, or at least that the image files are stored elsewhere and the protocols identifies an external method to physically retrieve them. The response table will also contain UCD information for the protocol defined fields to allow the clients to understand which fields contain which value.

3.1.4 Simple Spectral Access

The third *simple* protocol managed by VO-Dance and here described in a few words is the Simple Spectral Access Protocol (SSAP [10]). It basically requires the same positional information of the SIAP plus additional spectroscopic information for band coverage and time span of observation. Also the data retrieval is performed the same way the image protocol does, requiring a URI as the data access reference. What actually changes is the data modellization between the exposed datasets and the VO compliant output. This task was actually performed by UCDs in the two protocols seen above while in SSAP this is delegated to UTypes described in the document standard for the spectral data model.

The differences between these three protocols require some effort when trying to provide an engine capable of deploying all of them in a similar way, i.e. providing some abstraction to expose each of them as a particular flavour of the same access mechanism. This is what VO-Dance does, however some problems arise when trying to add to the picture also another main IVOA standard protocol that deals with generic database schemas, the TAP we describe in the next subsection.

3.1.5 Table Access Protocol

While the previous protocols deal with specific data types, catalogues, images and spectra, the Table Access Protocol [11] is meant to provide access to generic datasets contained in one or more database schemas. To achieve such a result a change of perspective is required because datasets may be large enough to require long time spans to be queried and also because results must be correctly described (again UCDs and UTypes are involved) to be interpreted by the clients consuming such a generic data service. The last issue can be solved if data models are provided, the first one means moving from the synchronous web services of the simple protocols to asynchronous resources for TAP. TAP is also more complex from the point of view of the service descriptors exposing the schema structure, query capabilities and results as a restful endpoint; this was not needed by the S*AP protocols above, which simply expose, if queried for metadata, the output fields they're capable of.

Database schema complexity and asynchronous behaviour are the two main reasons why at present VO-Dance cannot deal with TAP services, but before exploiting this point we describe the VO-Dance application.

4. VO-DANCE

On top of the archiving ingestion facilities of IA2, considering also the VO effort of the data center, a resource publisher was the next logic layer to put on. For the LBT user interface development, a web server was in charge to serve both the user interface and the VO clients, publishing SIAP and Cone Search services. For the other telescopes and instruments VO-Dance was a good choice to fit completely the target.

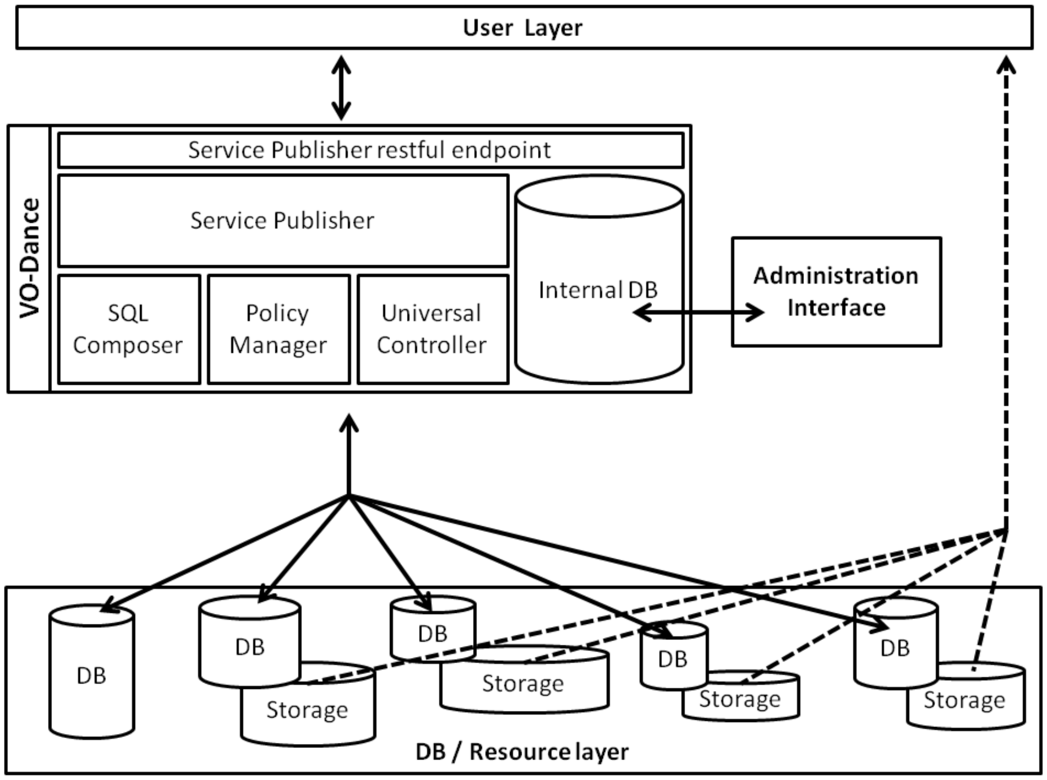

Figure 2. VO-Dance architecture.
User layer clients query services' endpoints managed by VO-Dance restful web application. Calls are taken in charge by the service publisher, and its companion components, and translated in SQL queries to the appropriate DB. The DB query response is formatted by VO-Dance and returned as an XML (VOTable) response to the initial client. The client can use the URI access reference from the VOTable to retrieve physical files from the DB connected storages.
Detached from this workflow the administration interface is connected to the VO-Dance's internal DB for services managament.

VO-Dance (see [12] and [13]) has been developed as an answer to the need of deploying in an easy and direct way the datasets ingested at the IA2 data center, datasets that are currently ingested using the DAS system (see 2.2). Requirements where set as: simple service maintenance, VO awareness and scalability with regard to new service protocols. The need for this tool is the natural consequence of an ingestion system like the DAS: characterize the data while ingesting it in an operative way and then distribute it by means of a simple transformation layer delined by some standard or protocol (in our case the VO protocols). VO-Dance deploys web services from already existing DB structures, eventually requiring only a new table or view to contain all VO needed data. This means the web application does not force in any way the archiving and ingestion process managed by the data centers.

In the following sections we will go through the VO-Dance architecture (Sec. 4.1), then we'll describe how metadata is characterized to fit the VO standards (Sec. 4.2) and finally we will show how the services are actually described and entered in the VO-Dance application (Sec. 4.3). Finally (Sec. 4.4) we will briefly deline the current VO-Dance status while future perspectives are exposed in the conclusions (Section 6).

4.1 Application architecture

VO-Dance is a Java Enterprise web application consisting in a set of Java libraries, a restful Java web interface and an internal DB schema to hold all the information needed to operate the resources' deployment. As depicted in Fig. 2 VO-Dance works as a translation layer between the DBs containing the resources to be deployed and the user layer's clients invoking the VO compliant services devoted to publish these resources.

The restful interface simply acts as a dynamic parametric endpoint that exposes the various services, collects the queries' inputs and passes them to the main application. The main application, i.e. the actual service publisher, is then fed with query parameters, instantiates the correct protocol driver and the service and invokes a JDBC connector to query to needed DB for results. Generating the correct query, applying (if it exists) a policy upon restricted access data and actually connecting to the DB is done by means of a set of independent packages, all integrating parts of VO-Dance. When the results from the DB resource are returned, the service publisher, with the aid of the information stored in the internal DB, formats them in the VO compliant form delined by the protocol in use and sends the XML/VOTable output back to the client.

The internal DB is connected to the main Java package by means of a persistence unit and entities that hold the DB schema structure for the application. The content of the internal DB consist of:

- external DB connection parameters: that allow VO-Dance to access resource information and retrieve the data queried by the clients;
- services general information: to identify and describe the services that VO-Dance dynamically generates and exposes with protocol identification;
- resource field mappings: to convert between DB columns on the original resources and VOTable fields in the query response (these include keys to identify UCDs, UTypes and other metadata);
- metadata information: to be connected to these fields to comply with the VO standard protocols implemented in the web application;
- log tables: for statistics on usage and for error handling.

Attached to the internal DB is also another application, usually called administration interface, that is actually responsible for ingesting resources descriptions, mapping information and, in general, all the content of the internal DB and its maintenace. In a few words the administration interface adds the services to VO-Dance while the VO-Dance web application itself is responsible for dinamically creating and exposing them.

The technologies used for VO-Dance make the application OS independent and, at least potentially, DBMS independent; however some requirements currently apply to its deployment:

- a Glassfish v2 web server is needed, mainly because of VO-Dance Entity Java Beans usage;
- MySQL and Oracle are the only strongly tested DBMSs for resources, and MySQL the only one for the internal DB (even if a working VO-Dance instance has been deployed using PostreSQL for the internal DB, P. Škoda private communication).

Working instances of VO-Dance run on Linux servers (CEntOS for the production site) and Windows (test and development instances), no attempt has been made to try to deploy VO-Dance on Mac OS X systems.

Of course not every token of VO-Dance has been coded from scratch but advantage has been taken from existing projects. From the VO point of view the libraries used to simplify VO-Dance development are the STIL (Star Table Infrastructure Library, [14]) library, for VOTable manipulations, and the Astrotime (CDS, Strasbourg, *http://cdsweb.u-strasbg.fr/*) library, mainly for time formats conversion.

The administration interface, which will be probably re-written in the near future, is based upon the python Django framework.

4.2 Metadata characterization

The previous subsection described the VO-Dance architecture, but a token is missing to understand how modelling metadata is added to existing resources to comply with VO standards. In this subsection we sketchily describe this point. i.e. we describe the data flux from the resource to the response VOTable when a service is invoked.

When a VO-Dance deployed service is invoked a protocol identifier and a service identifier are passed the service publisher togheter with query parameters and values. These information are used to retrieve the correct DB endpoint to query for (from the service id) and validate the query inputs against protocols requirements (using protocol identifier and query parameters). Then the query string is generated and used to query the DB schema to retrive the results.

At this point the web publisher performs a set of operations to format correctly the result to be returned to the client. The easier step is to convert the DB output table into a VOTable (this is done automatically using the STIL library). Another step requires adding the XML elements the metadata connected to the various fields of the VOTable; this is done using the information stored in the column mappers contained in the internal DB, that relate the DB table columns with VOTable fields. A less simple step is required when multiple DB table columns need to be merged in an array field of the VOTable.

Once all the trasformation and characterization steps are over, the XML response is ready to be sent back to the client via the restful interface which mediated the query call. For a Simple Cone search this actually closes the interaction between user and service. For SIAP and SSAP however there's also the possibility to retrieve the data from the URIs contained inside the VOTable response.

Since VO-Dance only connects to the resource's DB and the file transfer is actually demanded to the VOTable URI access reference for the files this means also that both the DB and the storage, that VO-Dance deploy as a VO service, can sit on geographically different servers from the one that hosts the VO-Dance web application. This capability can be useful for small projects that want to expose their resources through the VO without moving their DBs or files to the data center site (of course a DB access must be provided to the data center in this case).

4.3 Service deployment

As a last point before stepping into the current VO-Dance status, we describe what is needed to add a service to the VO-Dance web application and how to actually add it. Since VO-Dance, as refered by its name, is a VO related service publisher, VO description of the resource is needed to perform this operation.

Prior to this the resource must be fully contained in a single DB table or view, i.e. all needed fields required by the protocol one intends to use must be present in a single table. When this is done a DB user must be grant select permission to this table; user, password, DB connection will be stored in the internal DB of VO-Dance. These information are all what VO-Dance needs to connect to a DB. Of course also the VO protocol to be used, an eventual policy to be set, a name and a description for the service have to be input, as a minimum effort to identify the service.

These informations are added to the internal DB through the administration interface which presents on a single page form all the fields required to add the service. Of course these are only a small part of what is required, and indeed the bottom part of the form is left with the form fields needed to add the VO related metadata, i.e. the column mappers we introduced in the previous subsections.

The column mappers consist of a set of rows whose input are: the DB column name, the UCD, the UType and the Unit related to that column; one row for each exposed column that make up the resource we want to deploy as a VO service. After all the columns are mapped the user only needs to save the information into the VO-Dance internal DB and the newly created service is ready to be queried. The endpoint for the service is automatically generated from the server's URL, the web application context root, the protocol name and the service id.

4.4 Present status

At present the IA2 VO-Dance implementation must be considered in development phase, labelled as a 0.9 beta version. This does not mean that it is unstable or useless, it only means it requires refinement before stepping into a real maintenance phase. It can currently handle SCS, SIA and SSA protocols and actually it is in production use exposing a set of 19 published (VO registered) services and 5 test ones consisting in:

- 5 SIAP services: 3 TNG dedicated ones, 1 for the Tirgo/ARNICA instrument and a test one for the LBT lbc camera;
- 18 SCS services: 11 Planck ERCSC (Early Release Compact Source Catalog) dedicated, 4 services exposing SDSS extracted catalogues for candidate quasars and galaxies with related photometric redshifts (see [15]) and 3 generic test services;
- 1 SSAP service: actually active but not registered, using Ondrejov spectra (courtesy of P.Škoda) for implemetation test purposes.

As of this writing the set of 19 VO registered services are queried at an average rate of 1500 calls per week, globally.

Before describing which are the plans for the VO-Dance future developments we will brifley expose the present work aimed at adding TAP capabilities to this web application.

5. TAP TEST

As already stated in the TAP description subsection (Sec. 3.1.5) database schema complexity and asynchronous behaviour are the two main reasons why at present VO-Dance cannot deal with TAP services. This because the restful endpoint of VO-Dance only works for synchronous HTTP GET (or POST) calls and because the service publisher needs the deployed resource to be self contained in a single table or view. This means changes need to be done in the VO-Dance architecture to host also TAP implementation.

To understand in which way these changes can be performed while experiencing with asynchronous endpoints and the TAP implementation itself, a TAP attempt is in progress at the IA2 data center. This attempt relies upon the openCADC library developed by the Canadian Astrophysical Data Center (CADC, *http://www3.cadc-ccda.hia-iha.nrc-cnrc.gc.ca/cadc/*) that consists of a set of open source projects that give a good starting point for many VO standards implementation.

A working TAP service has been implemented and is actually used for test purposes and bug fixing, however the attempt has not given yet final results on which path to follow to add it to the VO-Dance web application.

6. CONCLUSIONS

In conclusion, at the IA2 data center efforts have been done and are going on to optimize archiving, ingestion and publication tasks when dealing with different astrophysical instrumentations. The DAS system is devoted to the first two tasks and has already been succesfully used for projects as big as the LBT-DA and TNG. The VO-Dance application is the answer to the last task and is currently in production for a set of registered VO services.

From the DAS point of view the *db_creator* will be upgraded to allow more complex DB schemas to be created, possibly with relational capabilities but also for databases following the NO SQL paradigm. Efforts will also be spent to try to generate a simplified mechanism for the creation of user interfaces, with pre-formatted form blocks and semi-automated interface generation.

Near future developments for VO-Dance will include a new administration interface, since the present one doesn't allow fine grain administration if multiple users are involved in administering resources (which requires row permission on internal DB while only table grants are available at DBMS level). Another step for VO-Dance will be the inclusion of the TAP protocol among those managed by the application, work on this development is in progress but will require major changes to the application architecture. Other changes will be made probably to include also planetary data models to those managed by the service publisher.

For the services, in general, a Single Sign-On addition is in a study phase to allow proprietary data to be published easily along with public data without the need for authentication at the level of the single service. We consider this of great importance especially when dealing with VO services because it will spread the VO paradigm over a larger community and, on the other hand, let the user take advantage of the VO tools even when dealing with their own data during the proprietary time span.

ACKNOWLEDGMENTS

We are pleased to acknowledge the work of Omar Laurino as the starting point for the VO-Dance development and that of Pietro Apollo for the TAP test service. Our work has also been simplified by the STIL library (Mark Taylor, Bristol University), the Astrotime library (developed by the CDS of Strasbourg, France) and the openCADC set of libraries (developed by the CADC VO group).

REFERENCES

[1] Knapic, C., Molinaro, M., and Smareglia, R., "From LBT to TNG: an easy way to inherit archiving system," *Astronomical Data Analysis Software and Systems XXI ASP Conference Proceedings* (in press).

[2] Knapic, C., Smareglia, R., Thompson, D., and Gredel, R., "LBT Distributed Archive: Status and features," *Astronomical Data Analysis Software and Systems XX ASP Conference Proceedings* **442**, 41–44 (2011).

[3] Gheller, C., Becciani, U., Costa, A., Manzato, P., Molinaro, M., and Pasian, F., "The Italian Theoretical VO data archive for simulated data," *Astronomical Data Analysis Software and Systems XVIII ASP Conference Proceedings* **411**, 135–139 (2009).

[4] Smareglia, R., Manzato, P., Gheller, C., Becciani, U., Manna, V., Marseglia, L., Pasian, F., and Taffoni, G., "Integration of theoretical data in the Virtual Observatory," *Astronomical Data Analysis Software and Systems XVI ASP Conference Proceedings* **376**, 587–590 (2007).

[5] Molinaro, M., Manzato, P., Gasparo, F., Pasian, F., Ameglio, S., Murante, G., and Borgani, S., "Preview, explore and analyze numerical simulations inside the VO," (2009).

[6] Manzato, P., Molinaro, M., Gasparo, F., Pasian, F., Pietrinferni, A., Cassisi, S., Rodrigo, C., Solano, F., and Cervio, M., "Micro-simulations inside the VO: the BaSTI case," *Astronomical Data Analysis Software and Systems XVIII ASP Conference Proceedings* **411**, 446–449 (2009).

[7] Ochsenbein, F., Williams, R., Davenhall, C., Durand, D., Fernique, P., Giaretta, D., Hanisch, R., McGlynn, T., Szalay, A., Taylor, M. B., and Wicenec, A., "IVOA Recommendation: VOTable format definition version 1.2," *ARXIV* **eprint arXiv:1110.0524** (2011).

[8] Martinez, A. P., Derriere, S., Delmotte, N., Gray, N., Mann, R., McDowell, J., Glynn, T. M., Ochsenbein, F., Osuna, P., Rixon, G., and Williams, R., "IVOA Recommendation: The UCD1+ controlled vocabulary version 1.23," *ARXIV* **eprint arXiv:1110.0518** (2011).

[9] Tody, D., Plante, R., and Harrison, P., "IVOA Recommendation: Simple Image Access specification version 1.0," *ARXIV* **eprint arXiv:1110.0499** (2011).

[10] Tody, D., Dolensky, M., McDowell, J., Bonnarel, F., Budavari, T., Busko, I., Micol, A., Osuna, P., Salgado, J., Skoda, P., Thompson, R., Valdes, F., and the Data Access Layer working group, "IVOA Recommendation: Simple Spectral Access protocol version 1.1," *ARXIV* **eprint arXiv:1203.5725** (2012).

[11] Dowler, P., Rixon, G., and Tody, D., "IVOA Recommendation: Table Access Protocol version 1.0," *ARXIV* **eprint arXiv:1110.0497** (2011).

[12] Molinaro, M., Laurino, O., and Smareglia, R., "Integrating the IA2 astronomical archive in the VO: the VO-Dance engine," *Astronomical Data Analysis Software and Systems XXI ASP Conference Proceedings* (in press).

[13] Smareglia, R., Laurino, O., and Knapic, C., "VO-Dance: VO Data Access Layer service creation made easy," *Astronomical Data Analysis Software and Systems XX ASP Conference Proceedings* **442**, 575–578 (2011).

[14] Taylor, M. B., "TOPCAT & STIL: Starlink Table/VOTable processing software," *Astronomical Data Analysis Software and Systems XIV ASP Conference Proceedings* **347**, 29–34 (2005).

[15] Laurino, O., D'Abrusco, R., and Riccio, G. L. G., "Astroinformatics of galaxies and quasars: a new general method for photometric redshifts estimation," *Monthly Notices of the Royal Astronomical Society* **418**, 2165–2195 (2011).