Astronomical Data Analysis Software and Systems XXV
ASP Conference Series, Vol. 512
Nuria P. F. Lorente, Keith Shortridge, and Randall Wayth, eds.

Taking Advantage of Cloud Solutions to Balance Requests in an Astrophysical Data Center

Marco Molinaro,[1] Francesco Cepparo,[1,2]
Ivan Scagnetto,[2] and Riccardo Smareglia[1]

[1]*Istituto Nazionale di Astrofisica - Osservatorio Astronomico di Trieste, via G.B. Tiepolo 11, 34143 Trieste, Italy;*
molinaro@oats.inaf.it

[2]*Università degli Studi di Udine, Dipartimento di Matematica e Informatica, via delle Scienze 206, 33100 Udine, Italy;*

Abstract. A complete astrophysical publishing environment, working as a helper system for an astrophysical data center, requires various components, from custom data back ends up to more or less standardized (e.g. Virtual Observatory driven) front end solutions. Combining this environment into one framework can lead to a potentially non scalable or hardly improvable system. In this contribution we describe what we are planning and developing to take advantage of cloud computing infrastructures and of a modular/distributed component architecture to provide a scalable and maintainable publishing environment at the Italian center for Astronomical Archives (IA2) at the INAF (Italian National Institute for Astrophysics) Astronomical Observatory of Trieste. Using a set of modular services, connected by registered interfaces, we are planning to use automated balancing at the front end to allocate services on demand in a cloud environment and allow generic data access in the back end archive solution.

1. Introduction

The IA2 data center serves its data in the form of Virtual Observatory (VO) services by means of a Java web application (named VODance, Molinaro et al. (2012)) capable of publishing IVOA (International Virtual Observatory Alliance) standard based Cone Search (Williams et al. 2011), SIA (Tody et al. 2011) and SSA (Tody et al. 2012) services. Over the last couple of years, due to changes in the IVOA Data Access Layer (DAL, see e.g. Bonnarel et al. (2016)) as well as in the available information technology solutions, this publishing system has been put under revision. The reasons for this revision are various: less uniform interface definitions for the various IVOA access protocols; monolithic web application approach; Java as the only available language for coding both the interface and the data access; maintainability of the application code; scalability of the publishing system; complex, RDB based, configuration of services.

Here we present the current status of the ongoing revision, from starting ideas to a first skeleton implementation of the new publishing solution.

2. Existing Architecture

Flexibility of protocol capability of the VO-Dance solution is based on the Java Reflection technology, allowing generic data access protocol classes to coexist in one web application. While this flexibility allows protocol-type based access classes to work on top of generic database back ends, the drawback sits in the overload of the reflection system and in its effect on interface high availability. Fig. 1 shows, in a sketchy format, the approach used within VO-Dance. It looks clear from the figure how the monolithic approach and the use of a web container limits the horizontal scalability of the system. Only independent implementations (and thus independent administrations and configurations) can allow for larger number of services when the load of the server becomes critical. This means higher maintenance costs.

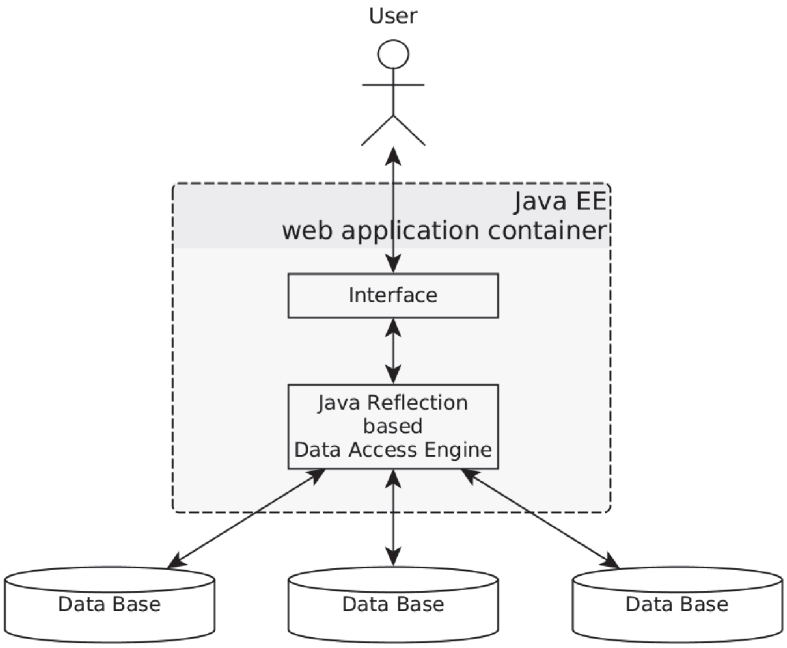

Figure 1. VO-Dance Architecture, skeleton diagram. Actually the *Interface* and the *Data Access* modules are separated only because they live in different packages with a dependency of the *Interface* on the *Data Access*. DBs can be generic ones, once a suitable SQL implementation is provided in the code, but Java is set as the only possible language for coding such differentiation.

3. Revisisted Architecture

A first idea for revisiting the publishing system was to take advantage of the Java EE web containers services plus EJB and JNDI technologies to cover horizontal scalability and modularity of the publishing solution (see e.g. Molinaro et al. (2014)). That idea would also have left the ability to postpone until the actual implementation phase the choice of the coding language for the real data access. However also this solution

would have left unsolved the issue related to a container with real horizontal scalability. Thus a second approach was undertaken, resulting in the modular and distributed architecture portrayed in Fig. 2, with no explicit container for the various modules of the publishing framework and free choice of coding languages for every module. The logical separation, already available in VO-Dance, between interface and access modules has been enhanced using a message broker to collect user requests and dispatch them to the correct access module. The addition of a reverse proxy allows a more flexible configuration of the front end while allowing horizontal scalability if request loads were to increase. Modularity for data access lets back end and coding choice be generic and horizontal scalability be achievable simply by registering the modules in the broker. All auxiliary modules (VOTable (Ochsenbein et al. 2011) serializers, logging system, etc.) can live in this new architecture as standalone modules attached to the main ones by means of message driven communication. Coding language will be a choice of the developer depending on the requirements of the single module.

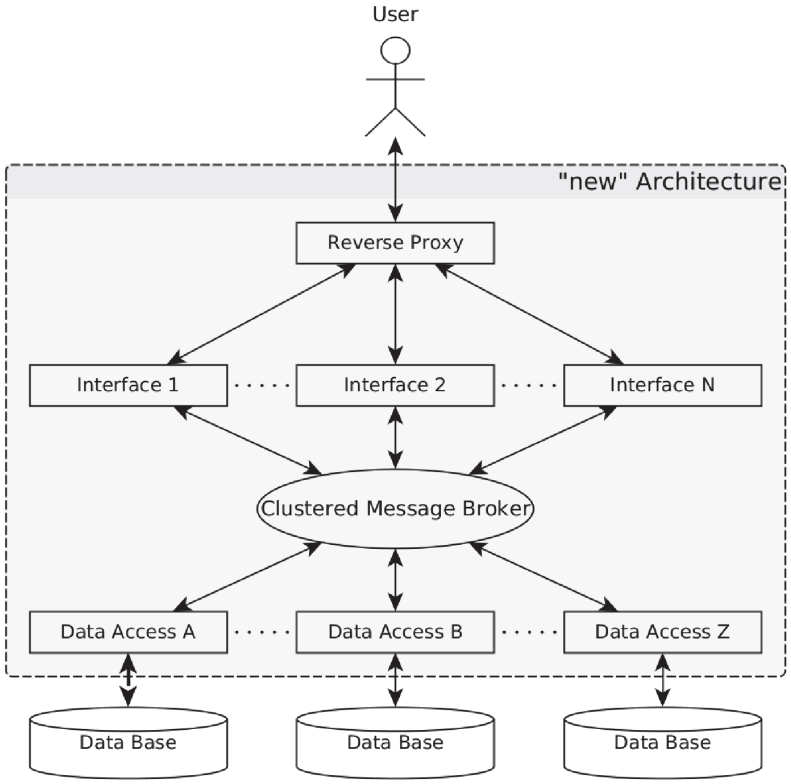

Figure 2. Revisited Architecture, skeleton diagram.

4. First Implementation

To prove the feasibility of the conceived solution a simple use case has been taken into account.

Two Cone Search services with separate interfaces and independent back ends have been implemented. While the interfaces in both cases have been realized using Java Servlet implementations, the two back ends have been kept separated both in terms of the database format and the coding language used to build the access module.

One of the two keeps a VO-Dance-like approach using Java together with a JDBC connection on top of a MySQL database table (it actually inherits all the business logic of the VO-Dance Cone Search classes and functions).

The other service is realized using an access module written in Python connected to an ASCII file in JSON format. The business logic for this module is built wrapping the needed code from the Simple Cone Search Creator[1]; also the generation of the JSON database has been done using that package on a sample catalogue already available at IA2.

The message broker uses RabbitMQ and the content of the exchanged messages, is serialized in JSON to neatly hold all the needed structured information.

To provide horizontal scalability at the user interface, a reverse proxy solution has been set up using HAProxy.

5. Next Steps

Apart from auxiliary modules (like dedicated VOTable response serialization, now built inside the data access modules, or a general logging module) future steps will involve porting into the new publisher the pre-existing IVOA protocol capabilities of VO-Dance. That will allow substitution of the current service publishing environment. In doing so service configuration would be added.

Acknowledgments. The technologies used for the implementation as well as for the new architecture and the actual development phase for the first implementation have been brought over by F. Cepparo as part of a degree thesis in informatics.

References

Bonnarel, F., et al. 2016, in ADASS XXV, edited by N. P. F. Lorente, K. Shortridge, & R. Wayth (San Francisco: ASP), vol. 512 of ASP Conf. Ser., 547
Molinaro, M., Knapic, C., & Smareglia, R. 2012, in SPIE Conf. Ser., vol. 8451, 5
Molinaro, M., et al. 2014, in SPIE Conf. Ser., vol. 9152, 0
Ochsenbein, F., et al. 2011. 1110.0524
Tody, D., Plante, R., & Harrison, P. 2011. 1110.0499
Tody, D., et al. 2012. 1203.5725
Williams, R., et al. 2011. 1110.0498

[1] https://github.com/tboch/Simple-Cone-Search-Creator